

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю
121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)
на тему:

«Вебплатформа реєстр пошуку загублених речей»

Виконала студентка групи ВТп-21-1
СЕРГІЄНКО Анастасія Сергіївна

Керівник роботи:
ФАНТ Микола Олександрович

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«02» серпня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СЕРГІЄНКО Анастасія Сергіївна**

Керівник роботи: **ФАНТ Микола Олександрович**

Тема роботи: **«Вебплатформа реєстр пошуку загублених речей»**,
затверджена Наказом закладу вищої освіти від **«01» серпня 2025 р., №270/с**

Вихідні дані для роботи: **розробка вебплатформи, яка дозволить користувачам створювати, переглядати та шукати оголошення про загублені та знайдені речі, з використанням сучасних вебтехнологій.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Фант О. М.	02.08.2025р.	02.08.2025р.
2	Фант О. М.	02.08.2025р.	02.08.2025р.
3	Фант О. М.	02.08.2025р.	02.08.2025р.

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновку та списку використаних джерел.

Текстова частина викладена на 51 сторінках друкованого тексту. Список використаних джерел містить 33 найменувань і займає 3 сторінки. В роботі наведено 33 рисунки та 9 таблиць. Загальний обсяг роботи – 63 сторінка.

У роботі поставлені основні завдання на розробку вебплатформи реєстру пошуку загублених речей. Для реалізації було проведено порівняльний аналіз аналогів. Також було обрано клієнт-серверну архітектуру, яка працює у зв'язці з сучасними підходами до веброзробки. Наведені основні сценарії використання системи, логіка взаємодії між модулями, діаграми даних та алгоритми роботи ключових модулів. Спроектowana база даних з використанням PostgreSQL. Клієнтська частина розроблена за допомогою фреймворку Next.js із використанням TanStack Query, а серверна частина за допомогою фреймворку NestJS.

КЛЮЧОВІ СЛОВА: NextJS, PostgreSQL, MantineUI, React Hook Form, GraphQL, ПЛАТФОРМА, ФРОНТЕНД, БЕКЕНД, БД.

					Житомирська політехніка.25.121.2.000 - ПЗ							
Зм.	Арк.	№ докум.	Підпис	Дата	«Вебплатформа реєстр пошуку загублених речей» Пояснювальна записка			Літ.	Арк.	Аркушів		
Розроб.		Сергієнко А.С.								2	63	
Керівник		Фант М. О.						Житомирська політехніка, група ВТн-21-1				
Резидент		Чижморя О.Г.										
Зав. каф.		Вакалюк Т.А.										

ABSTRACT

The explanatory note to the bachelor's final qualification work consists of an introduction, three sections, a conclusion and a list of sources used.

The text part is set out on 51 pages of printed text. The list of sources used contains 33 names and takes up 3 pages. The work contains 33 figures and 9 tables. The total volume of the work is 63 pages.

The work sets the main tasks for the development of a web platform for the registry of lost things. For implementation, a comparative analysis of analogues was conducted. A client-server architecture was also chosen, which works in conjunction with modern approaches to web development. The main scenarios of using the system, the logic of interaction between modules, data diagrams and algorithms for the operation of key modules are presented. A database is designed using PostgreSQL. The client part is developed using the Next.js framework using TanStack Query, and the server part using the NestJS framework.

KEYWORDS: NextJS, PostgreSQL, MantineUI, React Hook Form, GraphQL, PLATFORM, FRONTEND, BACKEND, DB.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						3

ЗМІСТ

РЕФЕРАТ.....	5
ABSTRACT	3
ВСТУП.....	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	9
1.1. ВИЗНАЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ.....	9
1.2. АНАЛІЗ АНАЛОГІВ ПРОГРАМНОГО ПРОДУКТУ.....	11
1.3. ВИЗНАЧЕННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	14
1.4. ОБҐРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ВИМОГИ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ.....	16
Висновки до розділу 1	18
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ РЕЄСТРУ ПОШУКУ ЗАГУБЛЕНИХ РЕЧЕЙ	20
2.1. ВИЗНАЧЕННЯ ВАРІАНТІВ ВИКОРИСТАННЯ ТА ОБ'ЄКТНО-ОРІЄНТОВАНОЇ СТРУКТУРИ СИСТЕМИ.....	20
2.2. РОЗРОБКА БАЗИ ДАНИХ СИСТЕМИ	23
2.3. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ АЛГОРИТМІВ РОБОТИ СИСТЕМИ	25
2.4. РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ ПЛАТФОРМИ	30
Висновки до розділу 2	36
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ.....	37
3.1. ПОРЯДОК ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ ПАРАМЕТРІВ СИСТЕМИ.	37
3.2. СТРУКТУРА ІНТЕРФЕЙСУ ТА КОРИСТУВАЦЬКА ВЗАЄМОДІЯ	38
3.3. ТЕСТУВАННЯ РОБОТИ ПРОГРАМНОГО КОМПЛЕКСУ	44
Висновки до розділу 3	46
ВИСНОВКИ	48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	49
ДОДАТКИ.....	52

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						4

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних.

ПЗ - програмне забезпечення

ГБ - гігабайт

МБ - мегабайт

RPC - Remote Procedure Call, дослівно: Віддалений виклик процедури

SSR - Server-Side Rendering, дослівно: Рендеринг на стороні сервера

UI - User Interface, дослівно: Користувацький інтерфейс

API - Application Programming Interface, дослівно: інтерфейс прикладного програмування

HTTP - Hypertext Transfer Protocol, дослівно: протокол передачі гіпертекстових документів

tRPC - TypeScript Remote Procedure Call

ORM - Object-Relational Mapping, дослівно: Об'єктно-реляційне відображення

UML - Unified Modeling Language. дослівно: Уніфікована мова моделювання

JWT – JSON Web Token

JSON – JavaScript Object Notation

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						5

ВСТУП

Актуальність теми. З огляду на зростаючу мобільність населення, активний ритм життя та високу цінність особистих речей, проблема їх втрати набуває особливої актуальності. Люди щодня пересуваються між містами, користуються громадським транспортом, відвідують громадські місця, що збільшує ризик загубити ключі, документи, телефони чи інші важливі предмети. У таких умовах стає очевидною необхідність у зручних цифрових інструментах, які б полегшували процес повернення загубленого.

Важливим аспектом вирішення цієї проблеми є створення спеціалізованої вебплатформи реєстру пошуку загублених речей. Така система надає користувачам можливість швидко розміщувати оголошення про втрату чи знахідку, отримувати сповіщення та взаємодіяти з людьми, які можуть допомогти у пошуку. Це значно скорочує час і зусилля, необхідні для повернення втраченого майна, адже платформа забезпечує зручну комунікацію між усіма учасниками процесу.

Запровадження подібної системи підвищує ймовірність успішного повернення речей, зменшує рівень стресу власників та формує культуру взаємодопомоги серед користувачів. Крім того, вона може стати важливим кроком до цифровізації повсякденних послуг і прикладом практичного використання сучасних технологій для вирішення соціально значущих проблем.

Об'єктом дослідження є технології для автоматизації процесів взаємодії між користувачами та системою, які забезпечують реєстрацію загублених і знайдених речей, їх публікацію та подальший пошук. Досліджується можливість оптимізації роботи вебплатформи з метою підвищення ефективності пошуку, зручності користування та забезпечення стабільної роботи при значному навантаженні й великій кількості одночасних користувачів.

Предметом дослідження є використання вебтехнологій для забезпечення інформаційних потреб предметної області.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	6
		Фант М.О.				

Метою є проектування та реалізація вебплатформи реєстру пошуку загублених речей, яка забезпечить ефективну взаємодію між користувачами, надаючи можливість швидкого створення оголошень про втрати чи знахідки та здійснення пошуку за визначеними критеріями. Система покликана забезпечити надійну роботу в умовах зростаючої кількості користувачів, підвищити зручність користування завдяки інтуїтивному інтерфейсу та сприяти оперативному відновленню втрачених речей шляхом формування централізованого реєстру.

Завданням випускної кваліфікаційної роботи бакалавра є:

- постановка основних задач системи та узагальнення її структури;
- аналіз існуючих вебсайтів на обрану тематику, визначення унікальності;
- визначення інформаційних потреб предметної області дослідження;
- розробка адаптивного інтерфейсу вебсайту за допомогою Next.js та супутніх бібліотек і фреймворків;
- визначити варіанти використання платформи
- проектування бази даних за визначеною предметною областю;
- розробка функціоналу та реалізація алгоритмів платформи;
- написання серверної логіки;
- розгортання платформи на віддаленому сервері;

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						7

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Визначення та постановка задачі

Метою даної кваліфікаційної роботи є створення вебплатформи реєстр пошуку загублених речей на базі сучасного фреймворку Next.js, що забезпечить високу продуктивність, зручний інтерфейс та адаптивний дизайн. Основне завдання полягає у створенні платформи, яка дозволить користувачам легко знаходити загублені речі та публікувати інформацію про знайдені предмети.

Вебплатформа реєстр пошуку загублених речей - це вебсервіс, який надає користувачам можливість створювати оголошення про загублені або знайдені речі, а також переглядати існуючі оголошення. Успішний сервіс має забезпечувати зручність у використанні, високу надійність, безпеку даних та стабільну роботу. Одним із прикладів успішних рішень у цьому напрямку є Lost & Found платформи, які функціонують у великих містах та міжнародних аеропортах, допомагаючи людям відновити загублене майно завдяки зручній навігації та інтеграції з різними системами оповіщення.

Вебплатформа реєстр пошуку загублених речей повинна включати кілька ключових модулів, які забезпечують різноманітні функції та можливості для користувачів. Модуль користувачів дозволяє реєструватися, авторизовуватися, редагувати профіль, а також керувати своїми оголошеннями. Модуль оголошень надає можливість додавати, редагувати та видаляти повідомлення про загублені або знайдені речі, а також переглядати детальну інформацію про кожне оголошення. Модуль пошуку забезпечує швидкий доступ до оголошень за різними критеріями, такими як місце втрати, категорія предмета та час публікації.

Усі ці модулі разом створюють зручний доступ до функцій платформи та допомагають користувачам ефективно взаємодіяти з системою для задоволення їхніх потреб.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	9
		Фант М.О.				

Для досягнення мети кваліфікаційної роботи будуть використані сучасні засоби розробки вебзастосунків, обрані з урахуванням їхньої ефективності, гнучкості та підтримки масштабованості. Фреймворк Next.js забезпечує серверний рендеринг (SSR) та статичну генерацію сторінок, що значно покращує продуктивність та швидкість завантаження вебзастосунку, а також сприяє кращій індексації контенту пошуковими системами. React, як популярна бібліотека для створення користувацьких інтерфейсів, дозволяє будувати компонентно-орієнтовану структуру застосунку та ефективно управляти станом компонентів, що підвищує зручність розробки та підтримку коду. Серверна платформа Node.js використовується для реалізації бекенд-логіки застосунку, забезпечуючи асинхронну обробку запитів та високу продуктивність при роботі з одночасними користувачами. Реляційна база даних PostgreSQL обрана для надійного зберігання та структурованого управління інформацією про товари, користувачів та замовлення, забезпечуючи цілісність та безпеку даних. Для контролю версій коду та організації спільної роботи над проектом застосовується система Git, яка дозволяє відслідковувати зміни, здійснювати паралельну розробку та підтримувати стабільність проєктного середовища.

В результаті повинна бути створено вебплатформа реєстр пошуку загублених речей. Важливою складовою є забезпечення надійності та безпеки, оскільки дана система буде обробляти персональні дані користувачів.

Основними етапами розробки вебплатформи реєстр пошуку загублених речей є послідовне створення дизайну вебсайту, побудова архітектури клієнтської та серверної частин, розробка системи авторизації й реалізація основного функціоналу, а також тестування.

Розробка дизайну починається зі створення загальної концепції додатку, включає проєктування UI-компонентів та оформлення сторінок. Далі здійснюється побудова архітектури клієнтської частини, що передбачає вибір фреймворку (наприклад, Next.js), формування структури проєкту та створення компонентів інтерфейсу користувача. Паралельно розробляється серверна

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						10

частина: обирається база даних (PostgreSQL), визначається її структура та створюється API для взаємодії з клієнтом.

Окремим етапом є впровадження системи авторизації. На цьому етапі реалізується реєстрація й вхід у систему, та впроваджуються механізми безпеки.

Подальша робота зосереджується на основному функціоналі. Це включає реалізацію логіки доступу залежно від ролей користувачів, організацію віддаленої бази даних для зберігання інформації, імплементацію бекенду з боку клієнта та налагодження взаємодії між серверною і клієнтською частинами.

Завершальним етапом стає тестування платформи, що охоплює як функціональні перевірки роботи системи, так і оцінку зручності та коректності графічного інтерфейсу.

Результатом виконання плану дій є створення вебплатформи реєстр пошуку загублених речей, за допомогою якого користувачі можуть легко шукати свої загублені речі та публікувати свої втрати чи знахідки. Це забезпечить зручність використання та високу продуктивність вебдодатку, що відповідає сучасним вимогам до даних систем.

1.2. Аналіз аналогів програмного продукту

На сьогодні існує не так багато вебсервісів чи мобільних додатків, що допомагають користувачам знайти загублені речі або повідомити про знайдені предмети. Такі платформи сприяють ефективній взаємодії між людьми, забезпечуючи можливість реєстрації знахідок, публікації оголошень про втрати, комунікації між користувачами та інтеграції з геолокаційними сервісами.

В ході пошуку існуючого програмного забезпечення було знайдено та проаналізовано такі сайти:

1. Сайт «Поверни!»

Посилання: <https://verni.com.ua/uk/>

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	11
		Фант М.О.				

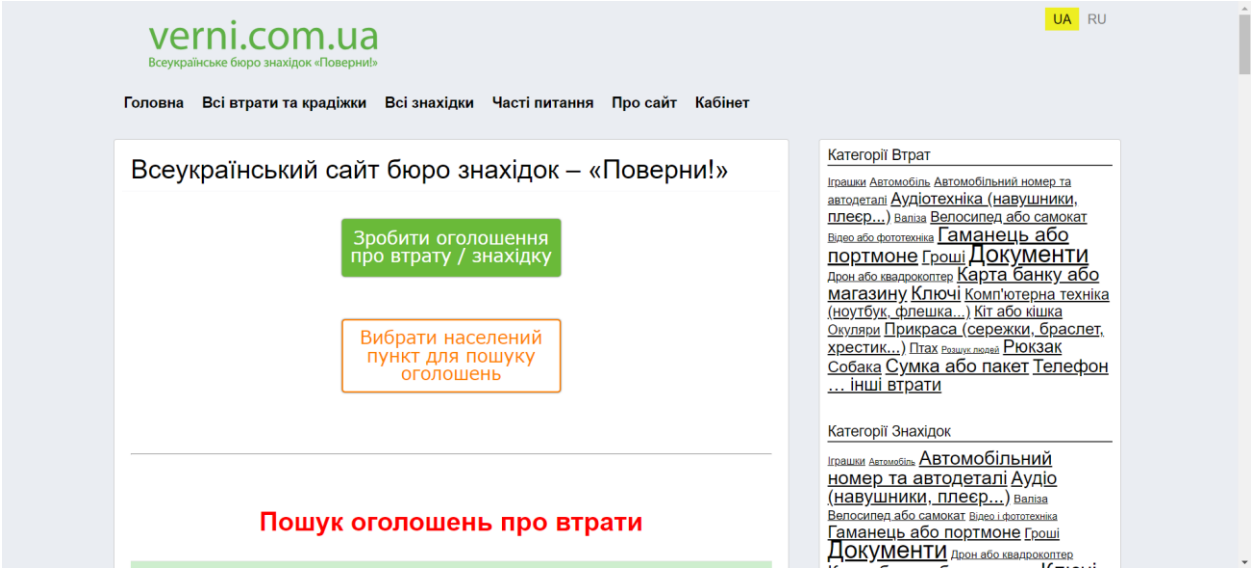


Рис. 1.1 Головна сторінка сайту verni.com.ua

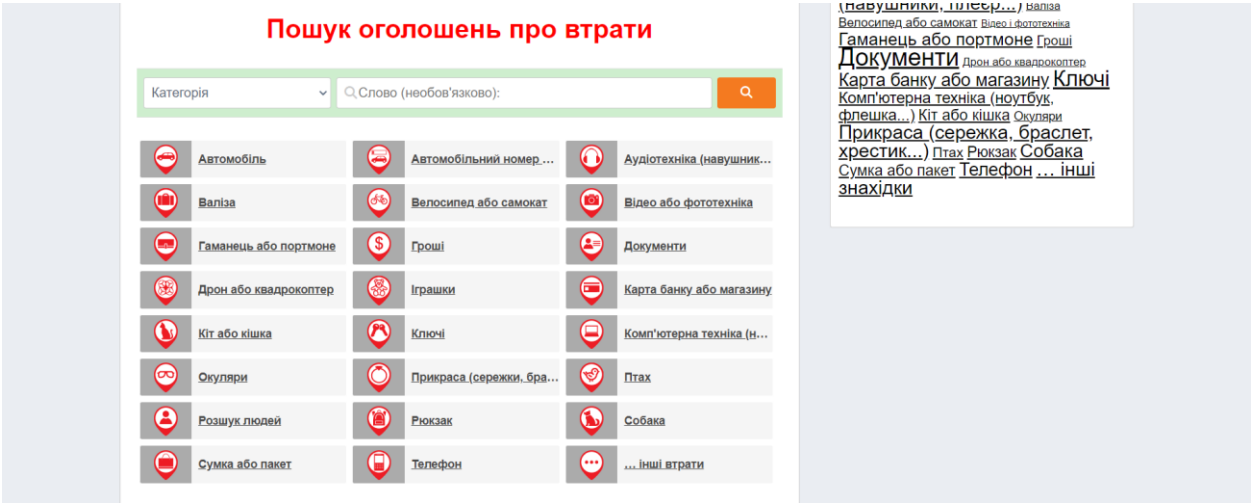


Рис. 1.2 Категорії для пошуку оголошень про втрату

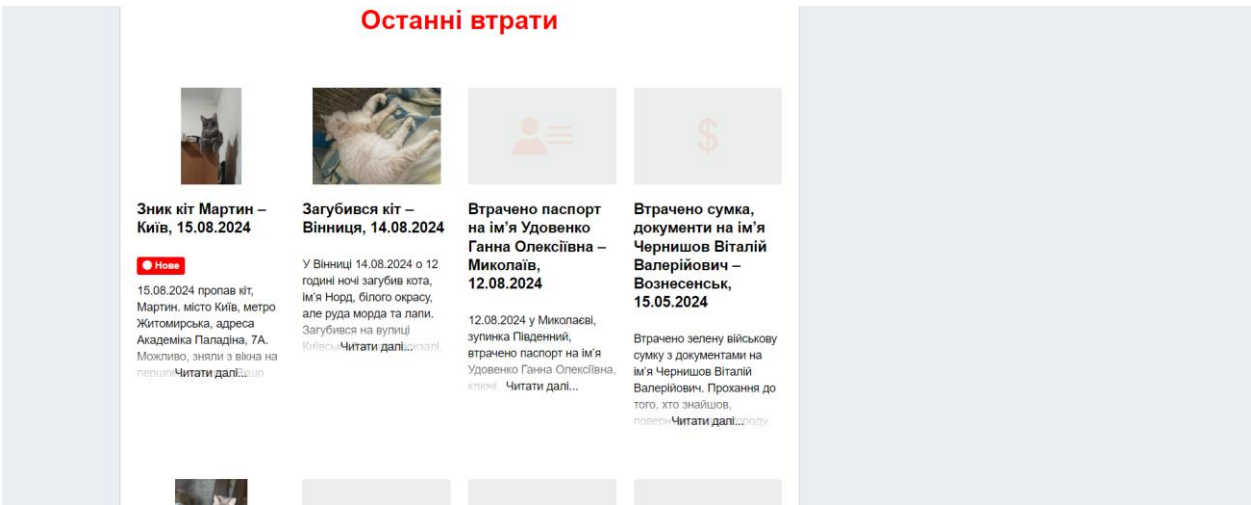


Рис. 1.3 Блок останніх витрат на сайті

Бюро знахідок в Україні «Поверни!» - це онлайн-сервіс для оголошень про знайдені та втрачені речі, документи й тварини. Дана платформа вирізняється зручністю у користуванні та швидким інтерфейсом, що робить його привабливим для користувачів. Водночас ресурс має і певні недоліки: дизайн виглядає недопрацьованим, на сторінках трапляються незрозумілі блоки, а також спостерігається надмірна кількість різних розмірів шрифтів в одному блоці, що ускладнює сприйняття інформації.

2. Сайт «LuckFind»

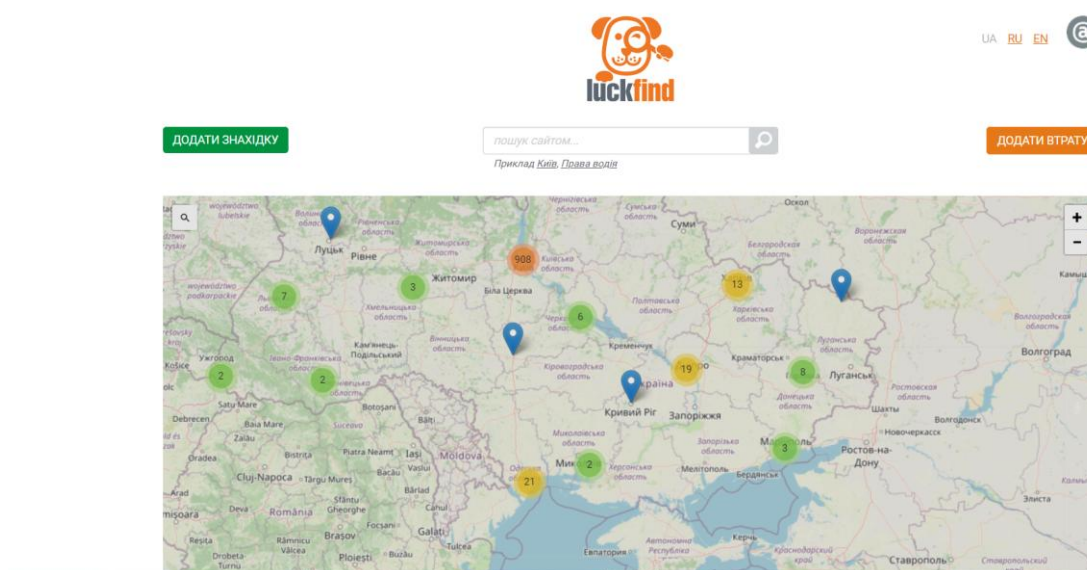


Рис. 1.4 Головна сторінка сайту LuckFind

LuckFind - це онлайн бюро знахідок, метою якого є знаходити та повертати втрачене. Дана платформа вирізняється зручним функціоналом для визначення локації, підтримкою кількох мов та мінімалістичним, але водночас зручним дизайном. Разом із тим, ресурс має певний недолік - слабку оптимізацію, що може впливати на швидкість роботи та зручність користування.

3. Сайт «Poisk.ua»

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	13
		Фант М.О.				

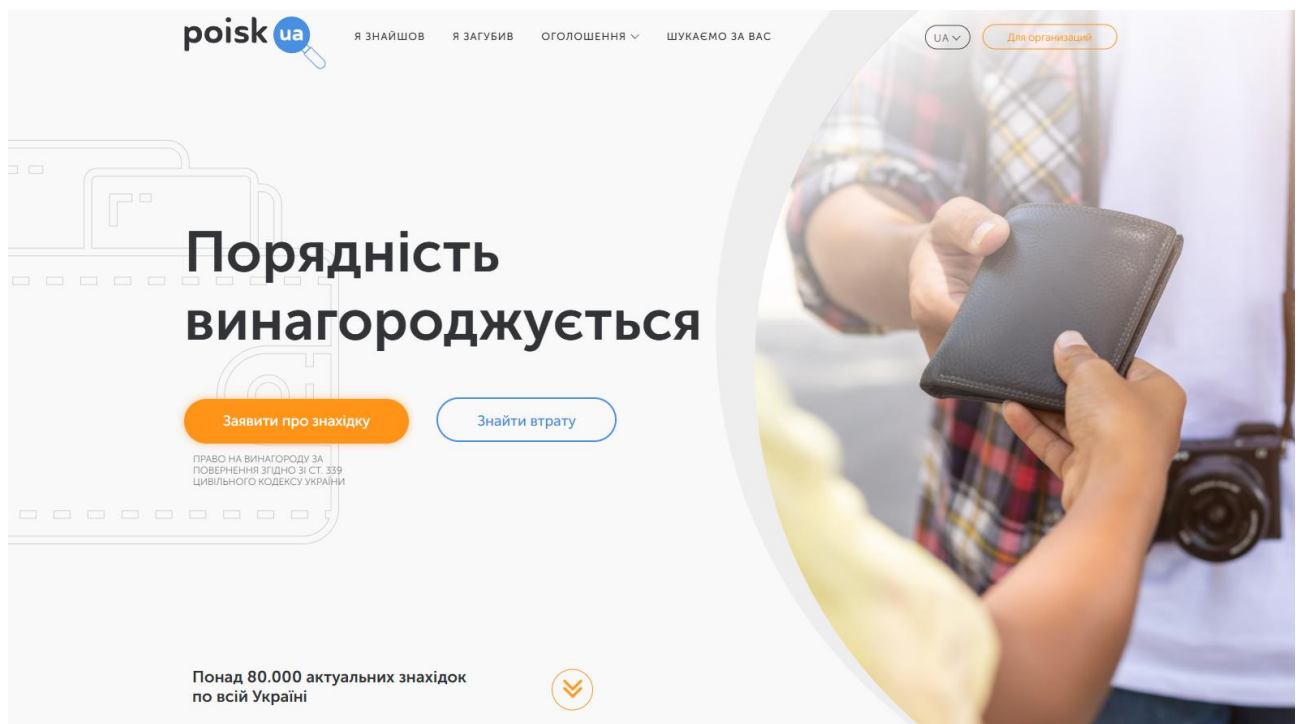


Рис. 1.5 Головна сторінка сайту Poisk

Платформа **Poisk.ua** надає можливість публікації оголошень про загублені та знайдені речі, а також пропонує автоматичний пошук і щоденні оновлення бази даних. Вона має простий інтерфейс та велику базу знахідок по всій Україні, що робить сервіс корисним для користувачів. Водночас існують і недоліки: функціонал для взаємодії між користувачами є доволі обмеженим, а мобільний застосунок повністю відсутній, що знижує зручність використання у сучасних умовах.

Отже, наша вебплатформа реєстр пошуку загублених речей має бути інтуїтивно-зрозуміле користувачеві, містити достатньо інформації про знахідки та мати зручний перегляд та сторення нових оголошень.

1.3. Визначення архітектури програмного забезпечення

Для розробки вебплатформи реєстру пошуку загублених речей було обрано клієнт-серверну архітектуру, у якій клієнтська частина взаємодіє із сервером для обробки, збереження та отримання даних (рис. 1.6). Завдяки цьому підходу забезпечено високу масштабованість, модульність і гнучкість функціональних можливостей платформи.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	14
		Фант М.О.				

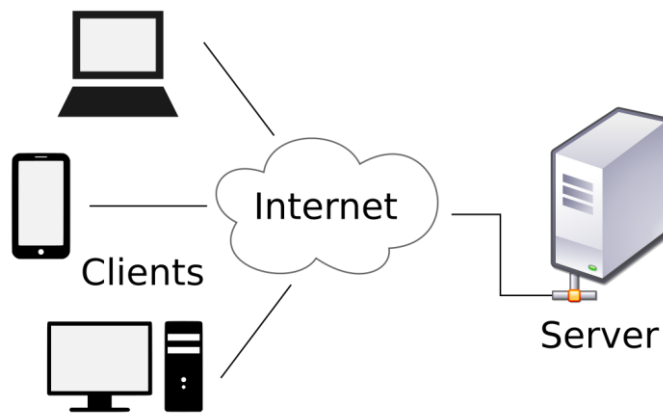


Рисунок 1.6 Схематичне зображення клієнтсько-серверного архітектури

Клієнтська частина вебплатформи розроблена з використанням Next.js, що забезпечує зручний та інтуїтивно зрозумілий інтерфейс, адаптований для різних пристроїв. Вона відповідає за відображення оголошень про знайдені й загублені речі, реалізує реєстрацію та авторизацію користувачів, а також взаємодію з серверною частиною. Для побудови інтерфейсу використовується бібліотека Mantine UI, яка надає готові сучасні компоненти та дозволяє швидко формувати адаптивний дизайн.

Система управління формами реалізована за допомогою React Hook Form, що забезпечує ефективне керування станом введених даних. Додатково застосовується бібліотека Zod, яка виконує валідацію та типізацію даних, підвищуючи надійність введеної інформації.

Серверна частина платформи побудована з використанням tRPC, що дозволяє реалізувати типобезпечну взаємодію клієнта та сервера без необхідності створювати окремий REST чи GraphQL API. Для зберігання оголошень, даних користувачів та категорій застосовується реляційна база даних PostgreSQL. Завантажені фотографії зберігаються у хмарному сервісі Vercel Blob, що забезпечує масштабованість та швидкий доступ до зображень.

Архітектура платформи ґрунтується на принципах модульності (modularization) та впровадження залежностей (dependency injection). Додаток складається з окремих компонентів, що полегшує додавання нових функцій та підтримку існуючої кодової бази. Використання dependency injection дозволяє

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	15
		Фант М.О.				

зменшити залежність між модулями, підвищує гнучкість конфігурації та спрощує тестування системи.

Завдяки такому підходу платформа є гнучкою, легко адаптується до змін, дозволяє впроваджувати нові функції без суттєвої перебудови всієї системи, а також забезпечує масштабованість і стабільність роботи.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки вебплатформи розглядалися кілька популярних середовищ розробки, серед яких Visual Studio Code (VS Code), PHPStorm та WebStorm. Вибір середовища був обумовлений потребою у зручному інструменті для фронтенд- та бекенд-розробки, інтеграції з системами контролю версій та базами даних, а також підтримки сучасних вебфреймворків.

Visual Studio Code є легким та швидким редактором з величезною кількістю розширень, що дозволяють додати підтримку Node.js, React, Next.js та інших технологій. Однак для повноцінної розробки вебплатформ він потребує налаштування та встановлення додаткових плагінів, що може знижувати продуктивність та вимагати додаткового часу на конфігурацію.

PHPStorm оптимізований для розробки на PHP та вебпроектів з використанням серверної частини PHP. Він забезпечує інтеграцію з Git, налагодження коду та тестування, проте менш зручний для сучасних JavaScript/TypeScript фронтенд-фреймворків і повної підтримки Node.js, що є важливим для реалізації розроблюваної платформи.

WebStorm спеціально розроблений для JavaScript, TypeScript та сучасних фронтенд-технологій, таких як React, Next.js, Angular і Vue. Він має вбудовані інструменти для налагодження, тестування та інтеграції з системами контролю версій і базами даних. WebStorm забезпечує оптимальну продуктивність при розробці вебплатформи, надає всі необхідні функції без зайвого перевантаження системи, що робить його найбільш ефективним вибором для даного проекту.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						16

Порівняння середовищ розробки

Параметр	Visual Studio Code (VS Code)	PHPStorm	WebStorm
Основна спрямованість	Легкий редактор з розширеннями	PHP та вебпроекти	JavaScript, TypeScript, сучасні фреймворки
Підтримка фронтенд-фреймворків	Через плагіни	Обмежена	Вбудована, повна
Підтримка Node.js	Через плагіни	Часткова	Повна
Налагодження коду	Через плагіни	Вбудоване	Вбудоване
Інтеграція з Git	Через плагіни	Вбудована	Вбудована
Продуктивність	Висока, легкий	Висока, але ресурсомісткий	Висока, оптимізовано для JS/TS
Простота використання	Потребує налаштувань	Простий для PHP, складний для JS/TS	Інтуїтивний для JS/TS, готовий до роботи
Вартість	Безкоштовно	Комерційна	Комерційна, з пробною версією

Після порівняння різних середовищ розробки(див. табл. 1.1) було вирішено використовувати WebStorm як основну IDE для реалізації платформи пошуку загублених предметів. Це середовище дозволяє швидко створювати та тестувати вебкомпоненти, працювати з серверною частиною на Node.js і підтримує сучасні стандарти веброзробки, забезпечуючи максимальну продуктивність та комфорт роботи.

Для організації зберігання даних та забезпечення стабільної роботи системи було проведено порівняння трьох популярних систем управління базами даних: PostgreSQL, MySQL та SQLite (табл. 1.2). Аналіз враховував такі параметри, як підтримка ACID, можливість роботи з форматами JSON, масштабованість, сфери використання та загальну продуктивність.

Порівняння БД

Параметр	PostgreSQL	MySQL	SQLite
Тип бази даних	Реляційна	Реляційна	Вбудована
ACID-підтримка	Наявна	Наявна	Наявна
JSON підтримка	Наявна	Наявна	Обмежена
Масштабованість	Висока	Висока	Низька
Використання	Веб, бізнес	Веб, CRM	Локальні БД

Після аналізу доступних варіантів для роботи з базами даних було вирішено використовувати PostgreSQL. Основними факторами вибору стали її надійність, гнучкість у роботі з даними та висока продуктивність.

Висновки до розділу 1

У першому розділі проведено детальний аналіз існуючих аналогічних платформ, визначено програмне рішення архітектури, продемонстровано раціональність вибору інструментів та сформульовано основні завдання розробки.

За допомогою аналізу було визначено ключові функціональні вимоги. Для цієї системи, зокрема створення каталогів знайдених і втрачених, інтеграція зручних особистих облікових записів користувачів, розробка ефективних механізмів пошуку та взаємодії між користувачами, для реалізації цього проекту була обрана клієнт-серверна архітектура, яка забезпечує масштабованість і гнучкість платформи. Клієнтська частина базується на Next.js, що дозволяє реалізувати адаптивний і зручний інтерфейс користувача. Для побудови UI використовується Mantine UI, а обробка форм здійснюється за допомогою React Hook Form у поєднанні з Zod, що забезпечує ефективну валідацію та типобезпечність введених даних. Для зберігання даних про користувачів, оголошення та категорії застосовується PostgreSQL. Використання сучасних

технологій дозволяє безпечно авторизувати, зберігати та обробляти дані в режимі реального часу.

Під час вибору технологічного стеку враховуються продуктивність, безпека, простота підтримки та можливість подальшого розширення. Впровадження модульної архітектури дозволяє гнучко змінювати та додавати функціональні можливості, не вимагаючи серйозних змін у базовій структурі.

Загалом обрана архітектура та технологія забезпечують високу продуктивність, надійність та масштабованість мережевої платформи, що сприятиме ефективному пошуку та поверненню втрачених речей.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						19

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ РЕЄСТРУ ПОШУКУ ЗАГУБЛЕНИХ РЕЧЕЙ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Система є вебплатформою для створення, редагування, публікації та пошуку оголошень про загублені й знайдені речі. Вона реалізована з використанням сучасних вебтехнологій, що забезпечують швидку роботу, надійність та зручність у користуванні.

Платформа базується на Next.js як основному фреймворку для фронтенду та серверного рендерингу, а також tRPC для типобезпечної взаємодії між клієнтом і сервером без написання додаткового API. Для зберігання даних про користувачів, оголошення та категорії використовується реляційна база даних PostgreSQL. Завантажені фотографії зберігаються у хмарному сервісі Vercel Blob, що забезпечує масштабованість і доступність файлів. Інтерфейс користувача розроблений за допомогою бібліотеки Mantine UI, яка надає сучасні компоненти для швидкої побудови інтуїтивно зрозумілого дизайну.

Обробка форм виконується за допомогою React Hook Form, що дозволяє ефективно керувати станом введених даних, а бібліотека Zod використовується для їхньої валідації та типізації, що гарантує правильність і безпечність введеної інформації.

Основними функціями системи є створення оголошень про загублені та знайдені речі із завантаженням фото, вибором категорії та зазначенням місця, а також пошук за ключовими параметрами. Передбачена можливість взаємодії користувачів через коментарі та повідомлення. У системі визначено дві основні ролі: Користувач і Гість (див. рис. 2.1).

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						20



Рис. 2.1 - Діаграма варіантів використання

Користувачі можуть зареєструватися або увійти до вже існуючого акаунту, переглядати оголошення про втрати та знахідки, створювати власні повідомлення із зазначенням потрібної інформації та фотографіями. Крім того, користувачі мають можливість змінювати дані свого профілю, зокрема контактну інформацію. Натомість гість має обмежений доступ, що передбачає лише перегляд каталогу втрат і знахідок та використання пошуку без можливості створення чи редагування власного контенту. Основні прецеденти використання платформи наведені у таблицях 2.1 – 2.7.

Таблиця 2.1

Опис прецеденту «Авторизація»

Назва	Авторизація
Опис	Вхід у систему за допомогою логіна та пароля для доступу до розширених можливостей.
Виконавці	Гість

Таблиця 2.2

Опис прецеденту «Реєстрація»

Назва	Реєстрація
Опис	Створення облікового запису з базовими персональними даними для подальшого використання системи.
Виконавці	Гість

Таблиця 2.3

Опис прецеденту «Перегляд втрат і знахідок»

Назва	Перегляд втрат і знахідок
Опис	Ознайомлення з усіма опублікованими оголошеннями без необхідності авторизації.
Виконавці	Гість, Користувач

Таблиця 2.4

Опис прецеденту «Редагування особистих даних»

Назва	Редагування особистих даних
Опис	Можливість змінювати інформацію у власному профілі (ім'я, контакти тощо).
Виконавці	Користувач

Таблиця 2.5

Опис прецеденту «Пошук предметів»

Назва	Пошук предметів
Опис	Використання фільтрів та ключових слів для пошуку загублених або знайдених речей.
Виконавці	Користувач

Таблиця 2.6

Опис прецеденту «Перегляд власних оголошень»

Назва	Перегляд власних оголошень
Опис	Перегляд усіх оголошень, які були створені поточним користувачем.
Виконавці	Користувач

Таблиця 2.7

Опис прецеденту «Додавання оголошення»

Назва	Додавання оголошення (втрата або знахідка)
Опис	Створення нового оголошення про загублений або знайдений предмет шляхом заповнення відповідної форми.
Виконавці	Користувач

2.2. Розробка бази даних системи

Для зберігання інформації про користувачів та публікації системи використовується база даних PostgreSQL, з якою бекенд частина Платформи спілкується за допомогою Prisma ORM.

Діаграма бази даних (рис. 2.2) відображає логічну структуру вебплатформи, призначеної для збереження та обробки даних, пов'язаних із пошуком загублених і знайдених речей, а також організацією комунікацій між користувачами. Центральним елементом моделі виступає таблиця User, яка зберігає унікальний ідентифікатор, ім'я, прізвище, електронну пошту, зашифрований пароль, графічне зображення профілю та дату підтвердження електронної адреси. Ця таблиця є базовою сутністю, що має зв'язки з усіма головними підсистемами.

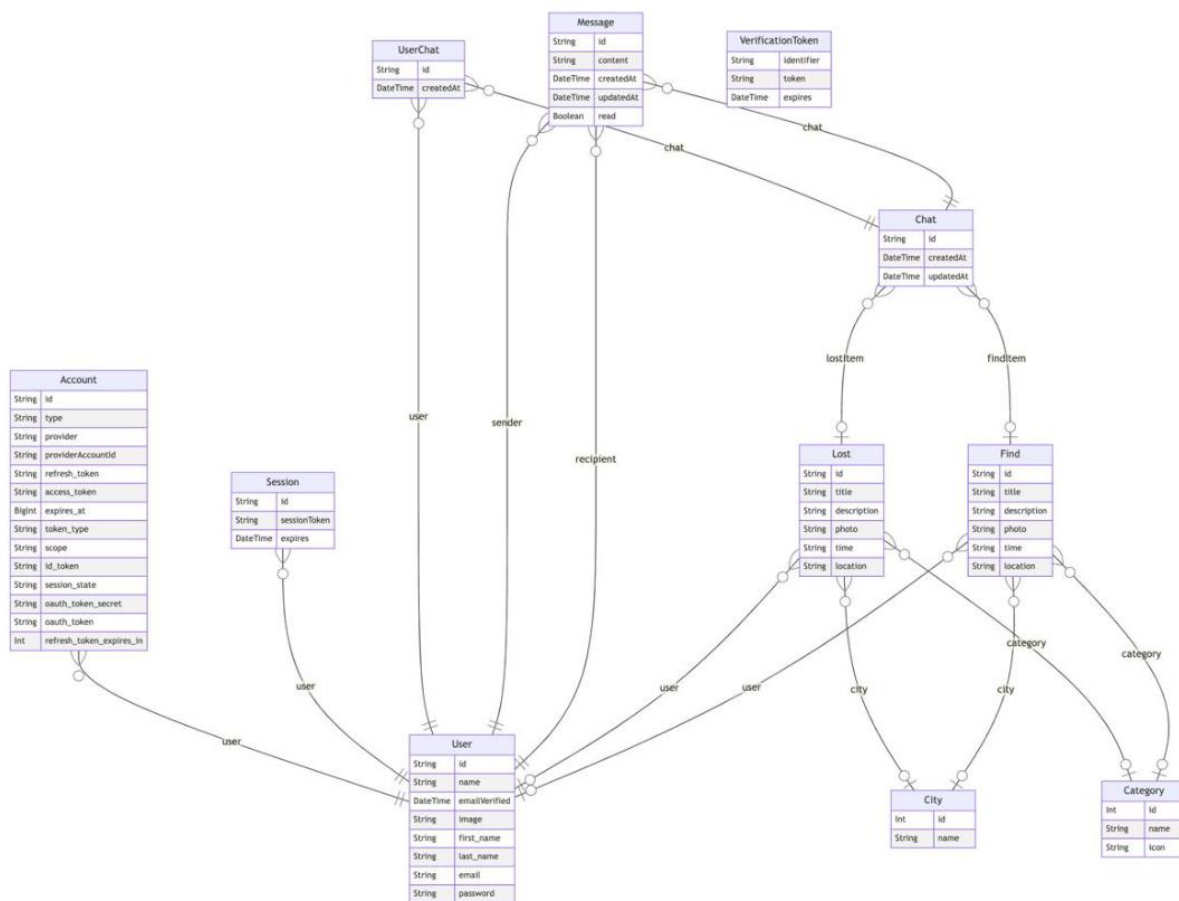


Рис. 2.2 Діаграма бази даних

Таблиця Account відповідає за зберігання інформації про зовнішні акаунти користувачів і параметри автентифікації, включаючи тип провайдера, набори токенів і строки їхньої дії. Для підтримки актуальних сесій використовується таблиця Session, де містяться унікальний токен доступу та час завершення його дії. Підтвердження електронної пошти реалізується через таблицю VerificationToken, у якій зберігається ідентифікатор користувача, токен підтвердження та дата його втрати чинності.

Механізм соціальної взаємодії між користувачами описується таблицею Chat, яка відповідає за створення та оновлення діалогів. Її доповнює таблиця UserChat, що визначає учасників конкретного чату, і таблиця Message, яка зберігає текст повідомлення, дату його створення й редагування, інформацію про відправника та одержувача, а також статус прочитання.

Ключовий функціонал системи побудований на роботі з оголошеннями, які представлені таблицями Lost та Find. Таблиця Lost зберігає дані про загублені

речі, тоді як Find містить інформацію про знайдені предмети. Обидві таблиці включають заголовок, опис, зображення, часову позначку, місце події та мають зв'язок із користувачем-автором запису. Для локалізації оголошень використовуються дані таблиці City із унікальним ідентифікатором і назвою міста, а для категоризації речей-таблиці Category, що містять назву категорії та графічну іконку.

Архітектура бази даних побудована таким чином, щоб забезпечити цілісність даних, захист доступу, ефективні механізми пошуку та класифікації оголошень. Вона також створює умови для безпечної й комфортної комунікації між користувачами системи.

2.3. Проєктування та реалізація алгоритмів роботи системи

В даному розділі буде побудована діаграма діяльності, яка є важливим інструментом на різних етапах розробки програмного забезпечення. Діаграми діяльності, схожі на блок-схеми, є популярними в інформаційних технологіях.

UML-діаграми діяльності дозволяють детально відобразити конкретні сценарії використання, представляючи потік дій у системі. Вони також можуть використовуватись для моделювання бізнес-процесів, де вони допомагають визначити послідовність подій та вимог.

Основні переваги та особливості використання діаграм діяльності полягають у тому, що вони можуть доповнювати будь-який елемент моделі з динамічною поведінкою та відображати дії, які відбуваються одночасно. Плавальні доріжки дозволяють розподіляти діяльності відповідно до ролей або об'єктів, які їх виконують, а траєкторії об'єктів показують зміни станів під час виконання діяльності. Складні діяльності можна деталізувати, розбиваючи їх на декілька дій та використовуючи вкладені діаграми. Діаграми діяльності також можуть застосовуватися для проєктування процесів або операцій, виступаючи як візуальна мова програмування.

На рис. 2.3 зображена діаграма діяльності, яка відображає основні дії користувача на платформі для реєстрації та пошуку загублених і знайдених

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						25

речей. Користувач починає з відкриття платформи, після чого він може зареєструватися чи авторизуватись. Якщо користувач не авторизований, він може лише переглядати вже опубліковані оголошення. Якщо авторизований, він має доступ до каталогу загублених і знайдених речей та власних оголошень.

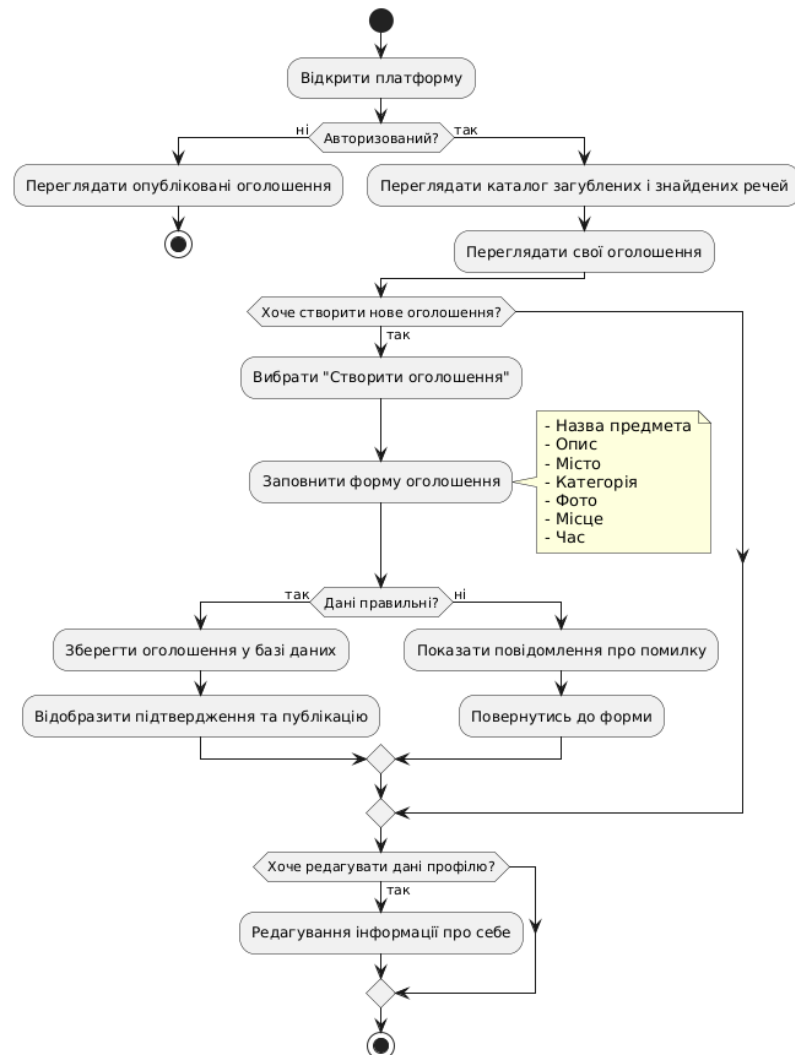


Рис. 2.3 Діаграма основного функціоналу платформи

Далі система визначає, чи хоче користувач створити нове оголошення. У разі позитивної відповіді користувач обирає опцію «Створити оголошення» та заповнює форму, що включає назву предмета, опис, місто, категорію, фото, місце та час. Після цього система перевіряє правильність введених даних. Якщо дані коректні, оголошення зберігається в базі даних і користувач отримує підтвердження про публікацію. У разі помилок система повертає користувача до форми для виправлення.

Також система може визначити, чи користувач хоче редагувати свій профіль. Якщо є така потреба, користувач може змінити інформацію про себе, після чого процес завершується.

На рисунку 2.4 показано діаграму послідовності, яка відображає процес авторизації користувача на платформі. Вона демонструє два сценарії входу: через обліковий запис Google та за допомогою логіна й пароля.

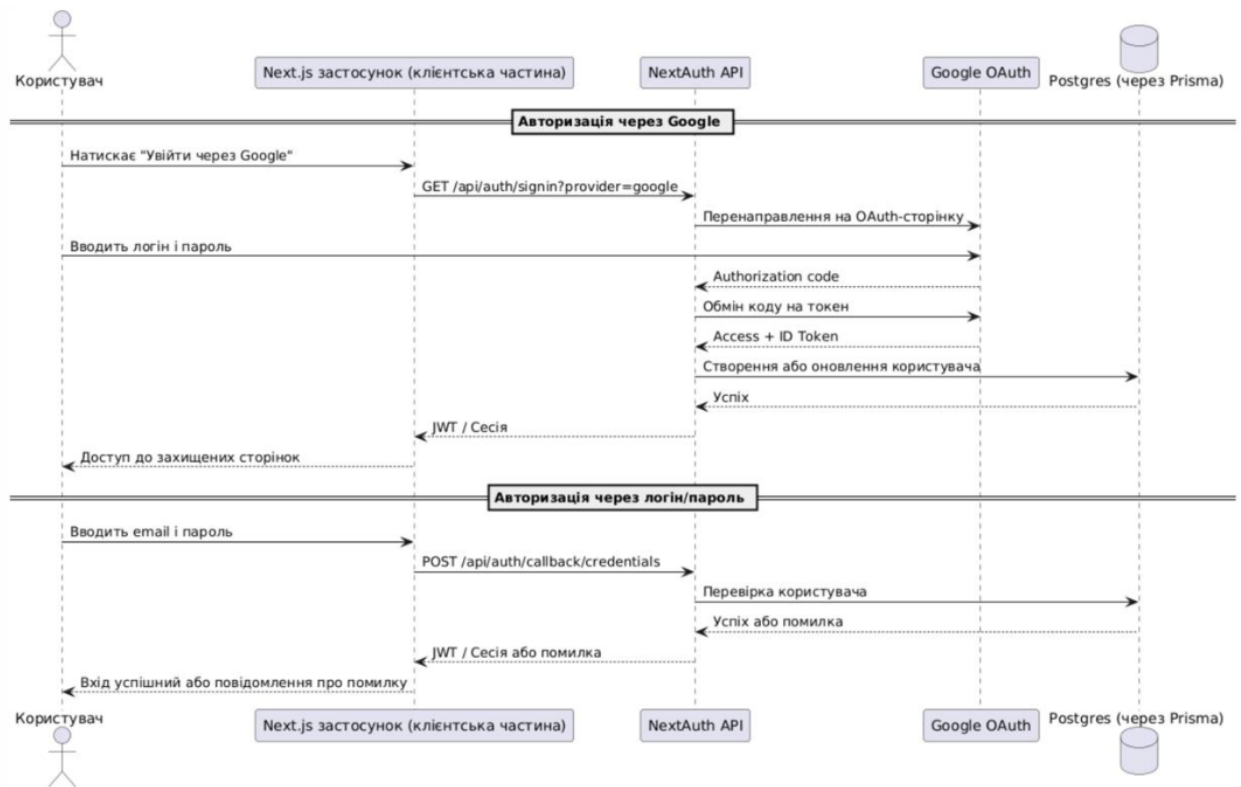


Рис. 2.4 Діаграма авторизації користувача

У першому сценарії користувач натискає кнопку «Увійти через Google». Після цього клієнтська частина Next.js ініціює запит до API автентифікації, що спричиняє перенаправлення на сторінку OAuth Google. Там користувач підтверджує вхід, і система отримує authorization code, який обмінюється на токени доступу та ідентифікації. Далі за допомогою Prisma ORM у базі даних Postgres створюється або оновлюється запис користувача. Успішна операція дозволяє NextAuth сформувати JWT або сесію, що відкриває доступ до захищених сторінок системи.

Другий сценарій передбачає авторизацію через логін і пароль. Користувач вводить дані для входу, які передаються за допомогою POST-запиту до API. Система звіряє введену інформацію з базою даних, і якщо перевірка проходить успішно, генерується JWT-токен або сесія для надання доступу до системи. У випадку некоректних даних користувач отримує повідомлення про помилку та можливість повторної спроби.

Діаграма показує, що система підтримує традиційну авторизацію за допомогою логіна й пароля, а також інтеграцію з зовнішнім провайдером Google OAuth, забезпечуючи високий рівень гнучкості та безпеки у процесі автентифікації.

На рисунку 2.5 показано діаграму основного функціоналу - пошуку загублених речей на платформі. Користувач починає з відкриття розділу «Пошук», після чого вводить ключові слова та обирає фільтри за категорією, датою та місцем.

Система отримує ці пошукові параметри і фільтрує оголошення у базі даних. Якщо результати знайдені, система відображає релевантні оголошення для перегляду користувачем. Якщо жодних результатів не знайдено, система показує повідомлення «Нічого не знайдено».

Після цього користувач переглядає результати пошуку, і процес завершується.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	28
		Фант М.О.				

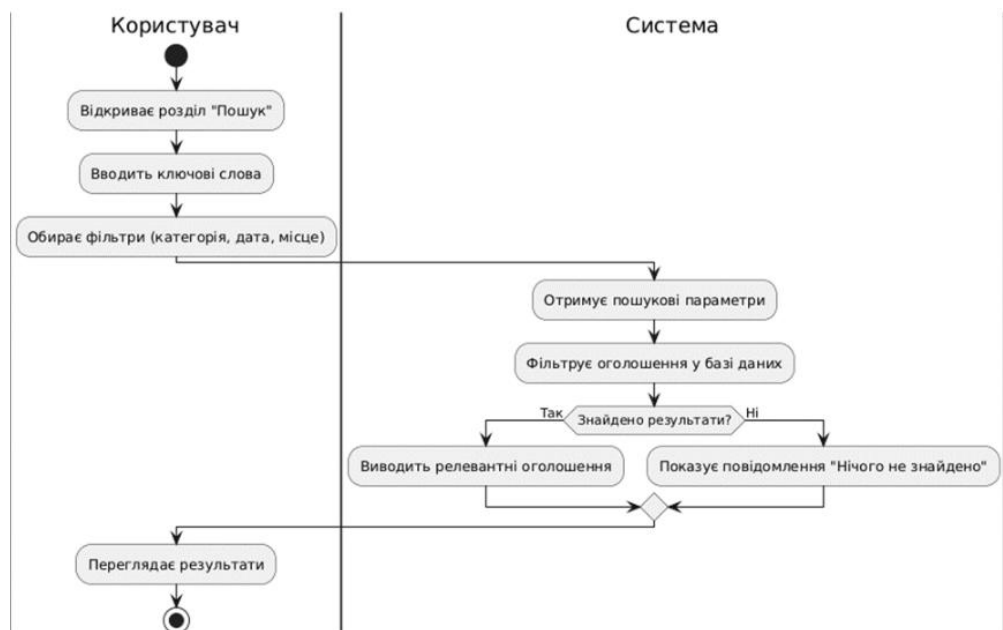


Рис. 2.5 Діаграма пошуку речей

Якщо користувач авторизований, він має змогу створювати власні оголошення. На рисунку 2.6 представлена діаграма послідовності, яка ілюструє процес розміщення оголошення про втрачену або знайдену річ на платформі. Діаграма поетапно показує взаємодію між користувачем та різними компонентами системи: клієнтською частиною, серверною логікою, ORM, базою даних і хмарним сховищем.

Процес починається з того, що користувач у клієнтському інтерфейсі Next.js вводить назву, опис і завантажує фотографію речі. Зображення передається до зовнішнього сховища Vercel Blob, яке генерує унікальну URL-адресу для цього файлу. Далі клієнтська частина формує tRPC-запит `lost.create()` до серверного роутера. Серверна логіка приймає цей запит і за допомогою Prisma ORM додає новий запис у базу даних Postgres, виконуючи операцію `INSERT` у відповідну таблицю. Успішне внесення даних підтверджується з боку бази та передається назад до клієнтської частини через сервер.

Після завершення операції система повідомляє користувача про успішне додавання оголошення. Згодом відбувається перенаправлення на сторінку, де відображаються всі раніше додані речі. Таким чином, діаграма демонструє повний цикл створення запису - від введення даних користувачем до їхнього успішного збереження в базі даних і отримання підтвердження.

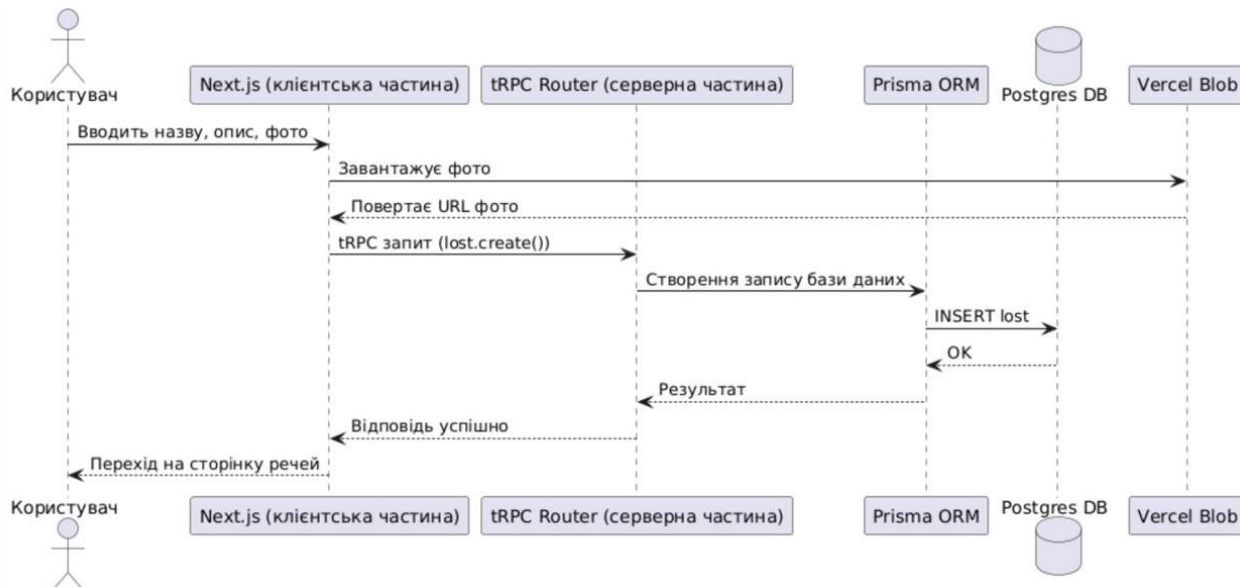


Рис. 2.6 – Діаграма для додавання втрачених та знайдених речей

2.4. Реалізація функціоналу платформи

Клієнтська частина реалізована з використанням фреймворку Next.js, який поєднує можливості серверного рендерингу та статичної генерації, забезпечуючи високу продуктивність та оптимізацію для пошукових систем. Такий підхід сприяє швидкому завантаженню сторінок і підвищує зручність користування платформою.

Для реалізації клієнтських запитів у системі застосовано бібліотеку tRPC, яка виступає типобезпечним прошарком між клієнтом та сервером. Цей сервер підтримує зв'язок між клієнтом і сервером без потреби у додаткових посередників на зразок REST або GraphQL. Важливою складовою TRPC є те, що він автоматично створює типи для вас, що означає, що ви не будете перемикатися під час перемикання між тим, про що просить клієнт і як ви насправді робите це на сервері.

Додатково, усі чутливі дані проходять валідацію за допомогою Zod-схем, що гарантує коректність введених користувачем значень та захищає систему від некоректних або потенційно небезпечних вхідних даних.

На рисунку 2.7 наведено приклад отримання всіх категорій разом із пов'язаними записами. У даному фрагменті коду показано використання tRPC у

поєднанні з ORM-системою Prisma, яка забезпечує високорівневу абстракцію для роботи з базою даних. Завдяки цьому реалізуються запити до таблиць із можливістю включення залежних об'єктів (наприклад, знайдених чи загублених речей).

```
export const categoryRouter : BuiltRouter<{ ctx: { prisma: PrismaClient... } > = createTRPCRouter({ Show usages  📄 sergienko-anastasiya
  getAll: publicProcedure.query(async ({ ctx } : ProcedureResolverOptions<{ prisma: Prisma... } > : Promise<({ losts: { id: string; descripti... => {
    return await ctx.prisma.category.findMany({
      include: {
        losts: true,
        finds: true,
      },
    });
  }},

  getById: publicProcedure.input(z.object({ id: z.number() })).query(async ({ ctx, input } : ProcedureResolverOptions<{ prisma: F
    return await ctx.prisma.category.findUnique({
      where: { id: input.id },
      include: {
        losts: true,
        finds: true,
      },
    });
  }},
}
```

Рис. 2.7 Код отримання всіх категорій

Також у процесі розробки клієнтської частини системи було застосовано набір користувацьких хуків(додаток А), що реалізують доступ до віддаленого API через бібліотеку tRPC. Використання хуків дозволяє інкапсулювати логіку взаємодії з сервером, спрощує повторне використання коду та забезпечує реактивне оновлення даних у разі їх зміни.

Завдяки цьому підходу розробник отримує можливість ефективно організувати роботу з основними сутностями системи - користувачами, загубленими речами, знайденими речами, категоріями та містами. Усі мутації супроводжуються автоматичною інвалідацією кешу, що гарантує актуальність відображених даних.

На рисунку 2.8 представлений хук, який забезпечує оновлення даних користувача на клієнті за допомогою бібліотеки tRPC. Після успішного виконання оновлення відбувається інвалідація кешу, що автоматично ініціює отримання актуальної інформації. Це дозволяє підтримувати узгодженість даних у всіх частинах застосунку без необхідності додаткових дій з боку користувача.

```

export function useUpdateUser() : UseTRPCMutationResult<{ first_name: strin... {
  const utils : TRPCContextPropsBase<BuiltRouter<{ ctx: {... = trpc.useUtils();
  return trpc.user.update.useMutation({
    onSuccess: () : void => {
      utils.user.getAll.invalidate();
    },
  });
}

```

Рис 2.8 Хук для оновлення даних

Одним з надважливіших модулів є модуль авторизації, що забезпечує класичну автентифікацію за електронною поштою та паролем, а також інтеграцію з Google через OAuth(див. додаток В).

Код налаштовує NextAuth для авторизації користувачів у Next.js з використанням Google OAuth та власних облікових даних через email і пароль. Користувач може увійти через акаунт Google або ввести email і пароль, які перевіряються у базі даних Prisma з порівнянням хешованого пароля за допомогою bcrypt. Для зберігання сесій використовується JWT, токени кодуються та декодуються бібліотекою jsonwebtoken з алгоритмом HS256(рис. 2.9).

```

jwt: {
  encode: ({ secret, token }: any) :any => {
    return jsonwebtoken.sign(token!, secret, {
      algorithm: 'HS256',
    });
  },
  decode: async ({ secret, token }: any) :Promise<JWT> => {
    const decodedToken :any = jsonwebtoken.verify(token!, secret, {
      algorithms: ['HS256'],
    });
    return decodedToken as JWT;
  },
},

```

Рис. 2.9 Кодування та декодування JWT-токенів

В обробці JWT додатково додаються спеціальні claims для Hasura, що визначають роль користувача та його ID, а об'єкт сесії поповнюється полем id користувача для доступу на клієнтській стороні. Використання prismaAdapter забезпечує інтеграцію NextAuth із базою даних через Prisma, а налаштування

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	32
		Фант М.О.				

теми і стратегії сесій робить інтерфейс адаптивним і безпечним. У цілому цей підхід дозволяє поєднувати безпечну аутентифікацію, інтеграцію з базою даних і підготовку токенів для подальшої авторизації у сторонніх сервісах.

Центральним інструментом управління серверною взаємодією виступає TanStack Query. Ця бібліотека реалізує патерн кешованих запитів (рис. 2.10), що дозволяє зменшити кількість повторних HTTP-запитів, забезпечити миттєвий ререндер інтерфейсу після змін на бекенді та синхронізувати дані у реальному часі. Дані фетчаться через функції, прив'язані до ключів запитів, що дозволяє гнучко контролювати інвалідовані записи, повторні отримання даних та оновлення кешу після мутацій.

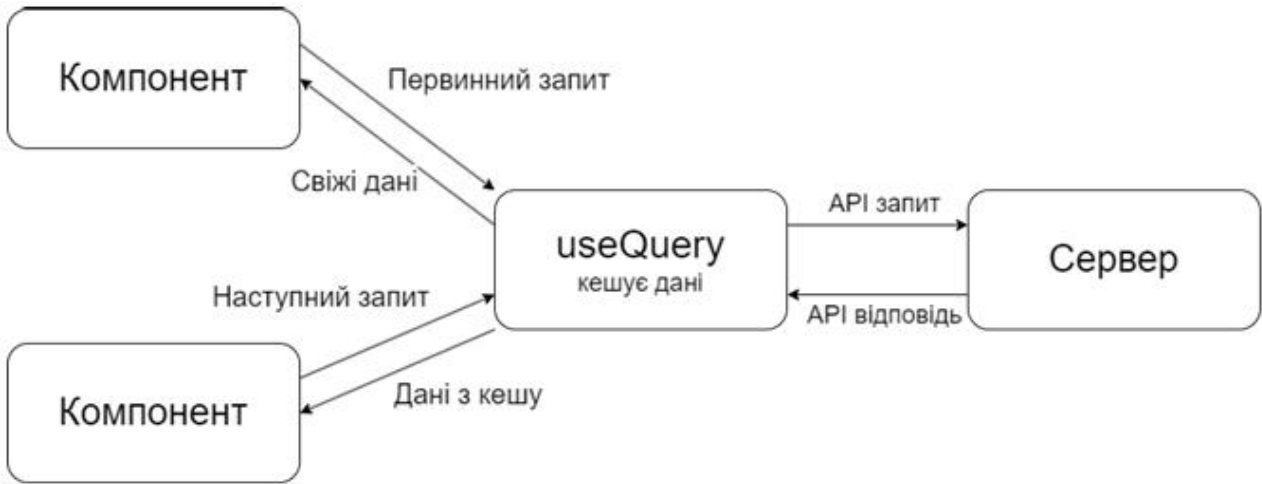


Рис. 2.10 Візуалізація роботи TanStack Query

Для оновлення даних також потрібно було створити спеціальну форму. На рисунку 2.11 показано фрагмент коду, який описує функцію, що відповідає за безпечне оновлення даних профілю користувача. Її робота ґрунтується на послідовному виконанні кількох етапів.

Спочатку функція виконує перевірку авторизації користувача. Це критично важливий крок для забезпечення безпеки, оскільки він запобігає несанкціонованим змінам даних. Якщо користувач не увійшов в систему, операція негайно припиняється, і з'являється відповідне повідомлення про помилку.

Якщо авторизація успішна, функція переходить до відправки даних на сервер. Це асинхронна операція, під час якої оновлені відомості про профіль (такі

як ім'я, прізвище, електронна пошта та URL аватара) надсилаються для збереження.

Завершальний етап - сповіщення користувача про результат. У разі успішного завершення операції з'являється підтверджувальне повідомлення. Якщо ж сталася помилка, наприклад, через проблеми зі з'єднанням або валідацією на сервері, користувач отримує відповідне попередження, що допомагає йому зрозуміти статус запиту.

```
const onSubmit: FormSubmitHandler<UpdateProfileForm> = async ({ data } : { data: { first_name: string; last_name: ... } : Promise<void> } => {
  if (!session?.user?.id) {
    notifications.show({
      title: 'Помилка',
      message: 'Користувач не авторизований',
      color: 'red',
    });
    return;
  }

  try {
    await mutation.mutateAsync({
      id: session.user.id,
      first_name: data?.first_name,
      last_name: data?.last_name,
      email: data?.email,
      image: data?.image,
    });

    notifications.show({
      title: 'Успіх',
      message: 'Профіль оновлено успішно',
      color: 'green',
    });
  } catch (error) {
    notifications.show({
      title: 'Помилка',
      message: error instanceof Error ? error.message : 'Помилка оновлення профілю',
      color: 'red',
    });
  }
};
```

Рис. 2.11 Код функції надсилення форми

Для створення інтерфейсу використовується бібліотека UI-компонентів Mantine, яка пропонує широкий набір готових і повністю кастомізованих елементів для розробки сучасних, адаптивних вебдодатків. Вона поєднує модульний підхід до побудови компонентів із гнучким і зручним API для їхньої стилізації.

```
"dependencies": {
  "@hookform/resolvers": "^3.9.0",
  "@mantine/core": "^7.11.2",
  "@mantine/dates": "^7.11.2",
  "@mantine/hooks": "^7.10.2",
  "@mantine/notifications": "^7.12.0",
```

Рис. 2.12 Підключення бібліотеки Mantine UI

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						34

Інтерфейс створюється за принципом складання складних елементів, таких як форми, навігаційні панелі чи картки, з базових компонентів Mantine. Наприклад, для реалізації форми публікації предмета можна використовувати такі компоненти, як TextInput, Select, Button, Group та Notification, що забезпечує узгодженість дизайну та функціональності.

Головна перевага Mantine полягає в її універсальності та швидкості розробки. Компоненти автоматично адаптивні, що гарантує правильне відображення на будь-яких пристроях. Крім того, бібліотека включає в собі систему тем і стилізації (Mantine Provider), яка дозволяє легко налаштовувати зовнішній вигляд усіх елементів у проєкті, підтримуючи єдиний візуальний стиль. Це дає змогу розробникам зосередитися на реалізації логіки програми без необхідності витрачати час на ручне написання стилів.

Основним елементом на сайті для пошуку загублених і знайдених речей є самі оголошення про речі. Для зручного перегляду кожної речі були створені спеціальні картки(рис 2.13), які дозволяють швидко ознайомитися з основними даними: назвою, описом, фотографією, місцем знаходження, часом та містом.

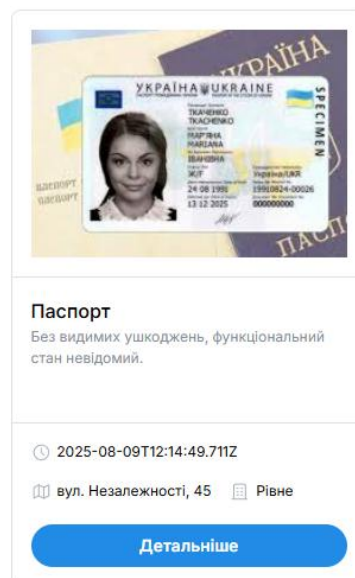


Рис. 2.13 Спеціальні картки для речей

У картках використано React для побудови інтерфейсу та динамічного оновлення даних, а TypeScript забезпечує строгий контроль типів і безпечну роботу з властивостями. Вони також містять кнопку для переходу до детального

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	35
		Фант М.О.				

опису речі. Детальний код створення таких карток можна переглянути у додатку Б.

Таким чином, картки забезпечують зручний та наочний спосіб перегляду оголошень, поєднуючи основну інформацію, взаємодію з користувачами та навігацію по сайту.

Висновки до розділу 2

В процесі написання другого розділу було розроблено діаграми варіантів використання, активності та компонентів. Досліджено варіанти використання та основний функціонал, який має бути наявний у додатку для користування платформою.

Також було спроектовано базу даних для зберігання даних, що використовуватимете платформа. Продемонстровано діаграму спроектованої бази даних, проведено опис кожної таблиці.

Завдяки побудові діаграм активності були окреслені ключові алгоритми взаємодії користувача із системою для подальшої їх реалізації. Це забезпечує чітке розуміння логіки платформи як для користувачів, так і для розробників.

В останньому пункті було успішно втілено основні функції системи, створивши повноцінний та функціональний програмний продукт.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						36

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ

3.1. Порядок встановлення та налаштування параметрів системи.

Система користувача повинна відповідати певним технічним вимогам для забезпечення повноцінної взаємодії з вебплатформою. Щодо апаратного забезпечення, мінімальною конфігурацією є процесор Intel Core i3 або AMD Ryzen 3 (або їх еквівалент), оперативна пам'ять обсягом не менше 4 ГБ та наявність вільного місця на диску не менше 500 МБ. Така конфігурація забезпечує стабільну роботу платформи та дозволяє виконувати основні функції без значних затримок або збоїв.

Щодо програмного забезпечення, користувач має використовувати сучасну операційну систему, таку як Windows 10, macOS 10.15 (Catalina) або більш нові версії. Важливим є також використання актуальної версії браузера - Google Chrome, Mozilla Firefox, Microsoft Edge або Safari, яка підтримує сучасні вебстандарти, зокрема HTML5, CSS3 та JavaScript ES6+. Це гарантує коректне відображення елементів платформи та підтримку всіх інтерактивних функцій.

Що стосується мережевого підключення, необхідно забезпечити стабільне інтернет-з'єднання зі швидкістю не менше 10 Мбіт/с. Таке підключення дозволяє безперешкодно передавати дані, швидко завантажувати сторінки та користуватися всіма функціями платформи в режимі онлайн.

Діаграма розгортання є важливим інструментом для візуалізації архітектури програмного продукту. Вона демонструє обчислювальні вузли, компоненти та об'єкти, які функціонують під час роботи програми. На цій діаграмі можуть бути відображені лише ті компоненти, які активно використовуються в процесі роботи програми.

На рисунку 3.1 зображено діаграму розгортання системи, яка демонструє архітектуру програмного продукту, що включає клієнтський браузер, сервер на базі Vercel, API та базу даних. Робота системи розпочинається з взаємодії користувача через браузер шляхом надсилання HTTP(S) запитів. Ці запити перенаправляються на сервер, розгорнутий на платформі Vercel, де функціонує

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						37

платформа, створена за допомогою Next.js. На сервері використовується tRPC для обробки запитів, що забезпечує виклик серверних функцій без необхідності використання традиційного REST API.

Застосунок на базі Next.js має прямий доступ до системи зберігання даних платформи Vercel Blob, що дозволяє працювати з великими файлами, включно зі збереженням та отриманням. Крім того, сервер взаємодіє з базою даних PostgreSQL, розміщеною в хмарному просторі Neon Cloud, використовуючи Prisma ORM для виконання SQL-запитів задля збереження та отримання інформації. Всі компоненти системи тісно пов'язані між собою, забезпечуючи її стабільне функціонування.

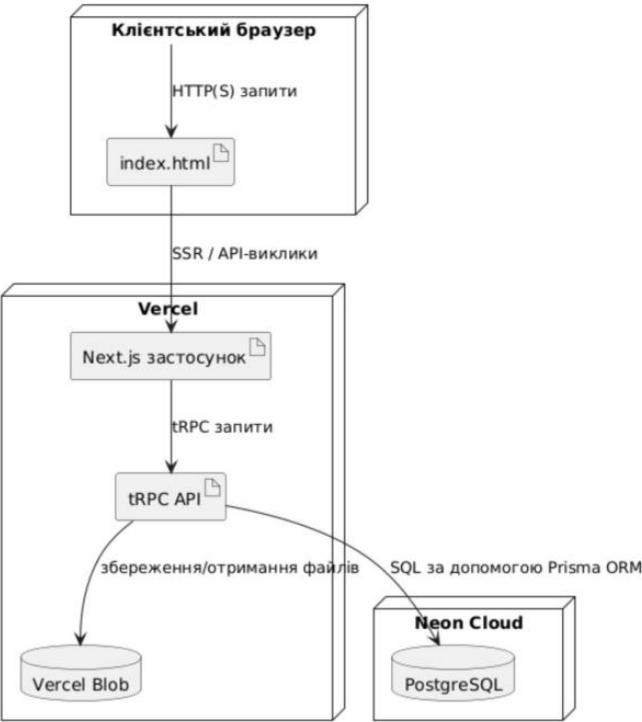


Рис. 3.1 Діаграма розгортання

3.2. Структура інтерфейсу та користувацька взаємодія

При першому візиті сайту неавторизований користувач бачить головну сторінку на якій користувачу потрібно обрати потрібну дію(рис. 3.2). В залежності від вибору користувачу буде відкрито список знайдених чи втрачених речей. Без авторизації можна переглянути весь список речей(рис. 3.3) та детальну інформацію про потрібне оголошення.

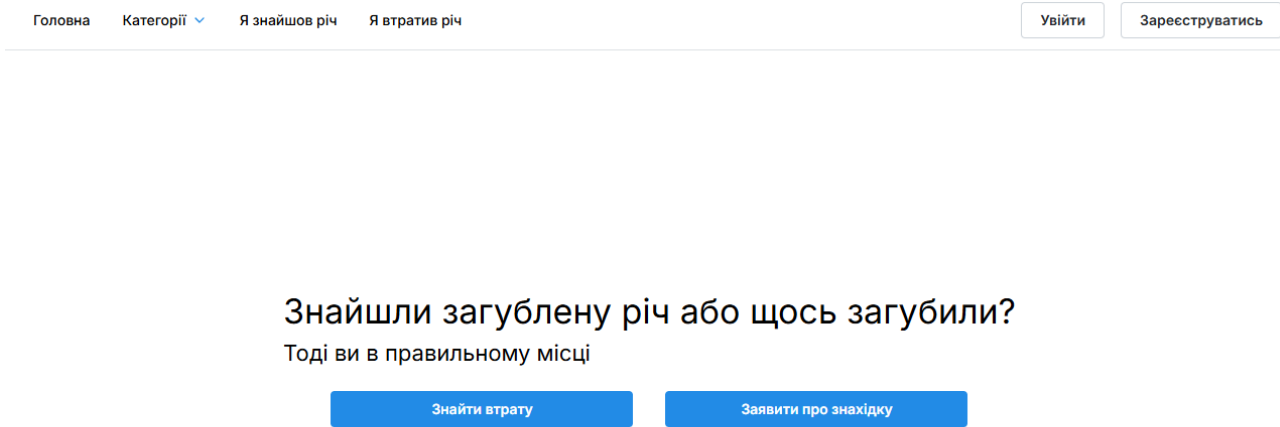


Рис. 3.2. Головна сторінка платформи

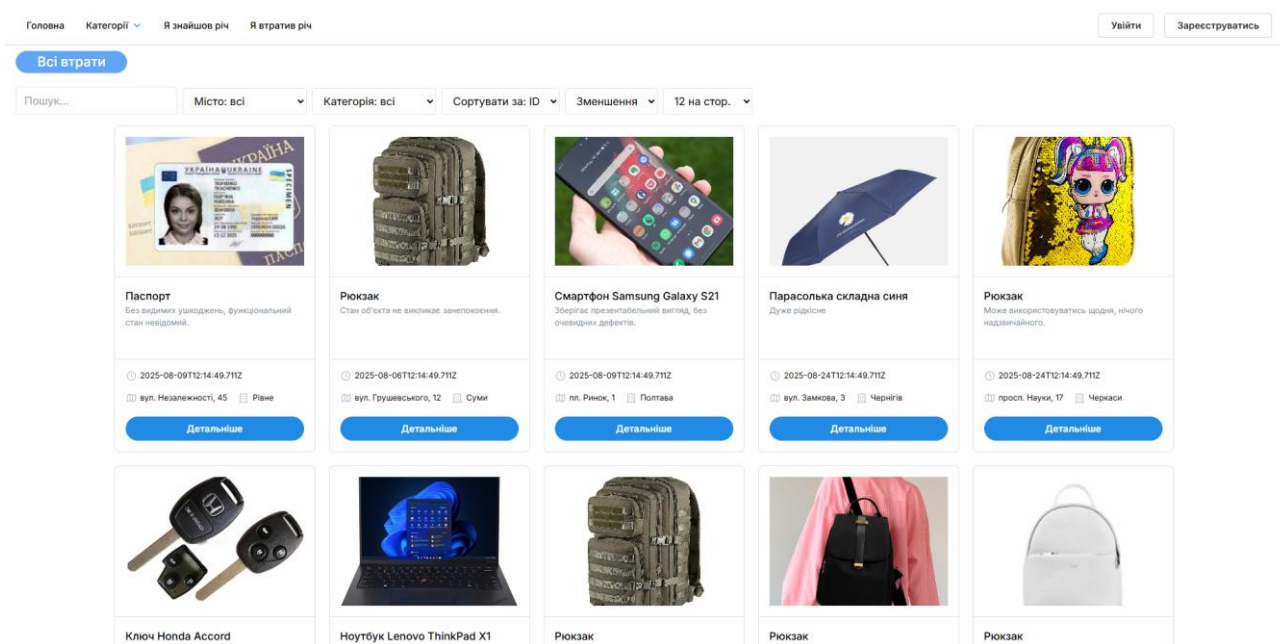


Рис. 3.3 Сторінка втрат

У верхній частині інтерфейсу(рис. 3.4) розташована горизонтальна панель навігації, яка надає користувачеві доступ до ключових розділів вебплатформи. Ця панель включає наступні елементи: «Головна», «Я знайшов річ» та «Я втратив річ».

Окремо можна виділити елемент «Категорії», який є випадаючим меню. Цей елемент розкриває нижню панель, яка містить кілька посилань для доступу до різних переглядів даних. Пункт «Втрати» дозволяє переглянути агрегований

список усіх оголошень про загублені предмети, тоді як «Мої втрати» надає персоналізований список оголошень про втрачені предмети, створених поточним користувачем. Аналогічно, «Знахідки» демонструє агрегований список усіх оголошень про знайдені предмети, а «Мої знахідки» відображає персоналізований список знайдених речей поточного користувача. Пункт «Повідомлення» забезпечує можливість спілкування авторизованих користувачів між собою.

Представлена структура інтерфейсу забезпечує інтуїтивно зрозуміле та логічно структуроване керування доступом до даних, розрізняючи загальну інформацію та персональні записи користувача.

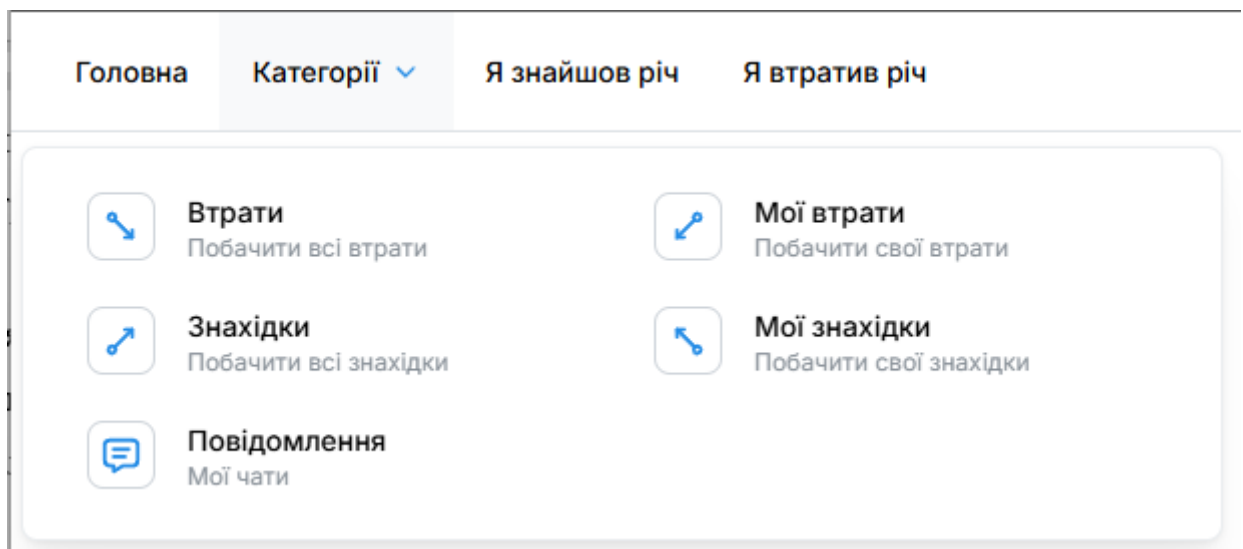


Рис. 3.4. Навігаційна панель платформи

На рисунку 3.5 та 3.6 зображено інтерфейс повідомлення, що відображається для неавторизованих користувачів під час спроби додати інформацію про втрату чи знайдену річ. У центральній частині вікна міститься заголовок та текстове повідомлення з вимогою авторизації: «Будь ласка, увійдіть в обліковий запис щоб додати втрачену річ/знахідку». У правому верхньому куті інтерфейсу розташовані кнопки «Увійти» та «Зареєструватись», що забезпечують можливість входу до існуючого облікового запису або створення нового. Така реалізація дозволяє обмежити доступ до функціоналу створення оголошень лише зареєстрованими користувачами, що сприяє підвищенню рівня достовірності опублікованої інформації.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	40
		Фант М.О.				

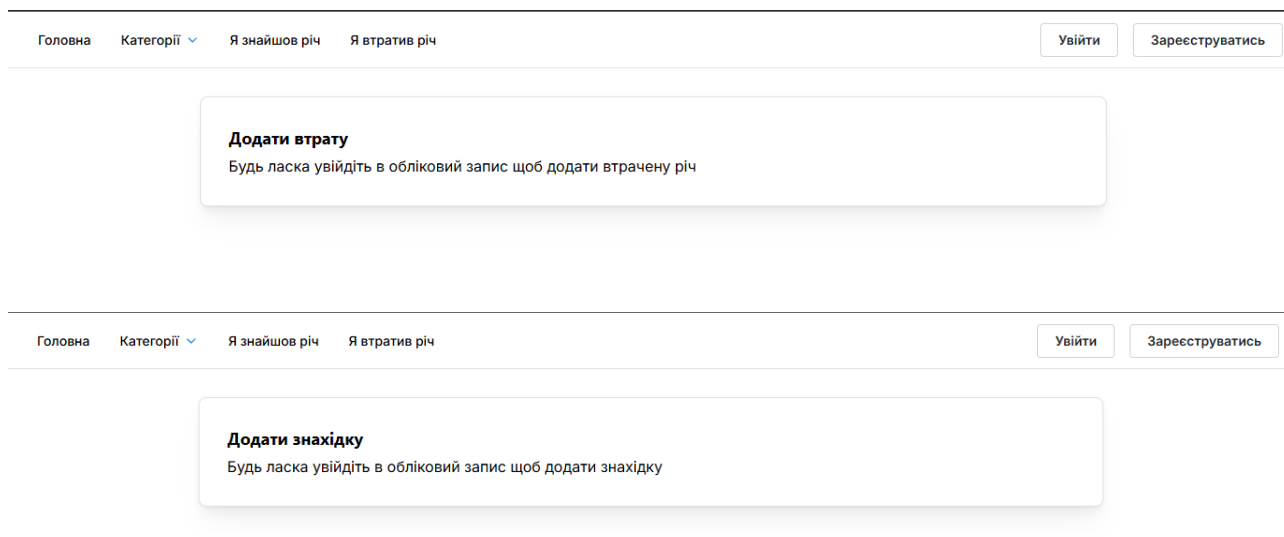


Рис. 3.5 – 3.6 Навігаційна панель платформи

На рисунку 3.7 зображено інтерфейс форми авторизації користувача в обліковому записі. У верхній частині розташована кнопка «Sign in with Google», що надає можливість виконати вхід за допомогою облікового запису Google. Нижче представлено альтернативний варіант авторизації шляхом введення електронної пошти та пароля. Поле «Email» містить приклад формату адреси (*example@example.com*), а поле «Password» призначене для введення пароля користувача. Така реалізація забезпечує гнучкість і зручність авторизації, дозволяючи користувачеві обрати найбільш прийнятний спосіб доступу до системи.

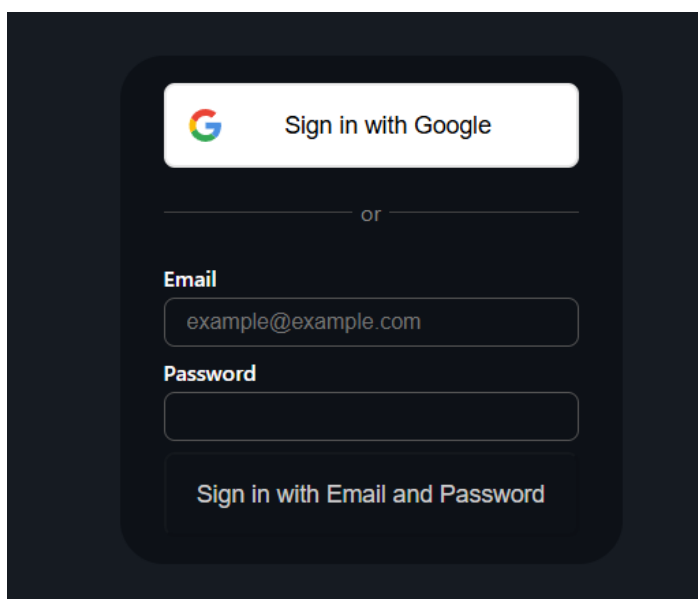


Рис. 3.7 Інтерфейс авторизації користувачів

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	41
		Фант М.О.				

Після авторизації користувач може додати оголошення про втрату чи знайдену річ. На рисунку 3.8 зображено інтерфейс форми для додавання знайденого предмета в систему. Центральний елемент складає форма, що містить низку полів для введення даних. Зокрема, користувач має можливість ввести назву об'єкта (обов'язкове поле), надати його опис, вибрати місто та категорію (обидва поля також є обов'язковими). Додатково реалізовано можливість завантаження фотографії, зазначення місця та часу, що сприяє більш детальній ідентифікації знахідки.

У нижній частині форми розташована кнопка «Опублікувати», яка виконує функцію збереження введеної інформації та її розміщення в базі даних платформи. Така структура інтерфейсу забезпечує простоту використання й дозволяє користувачу швидко додати інформацію про знайдений предмет.

The screenshot shows a web application interface for adding a find. At the top, there is a navigation bar with links: 'Головна', 'Категорії', 'Я знайшов річ', and 'Я втратив річ'. On the right, the user's profile 'nastiusha2623 solo' is displayed. The main content area features a form titled 'Додати знахідку'. The form includes the following fields: 'Назва' (required), 'Опис', 'Місто' (dropdown menu with 'Оберіть місто'), 'Категорія' (dropdown menu with 'Оберіть категорію'), 'Фото' (file upload button 'Оберіть файл'), 'Місце', and 'Час'. A blue 'Опублікувати' button is located at the bottom of the form.

Рис.3.8 Сторінка для додавання оголошення

Після додавання знахідки чи втрати влісні оголошення можна переглянути в вкладці «Мої втрати» або «Мої знахідки». Також можна переглянути детальну інформацію оголошення. Якщо авторизований користувач є власником відкритого оголошення, в нього є можливість видалити його(рис. 3.9).

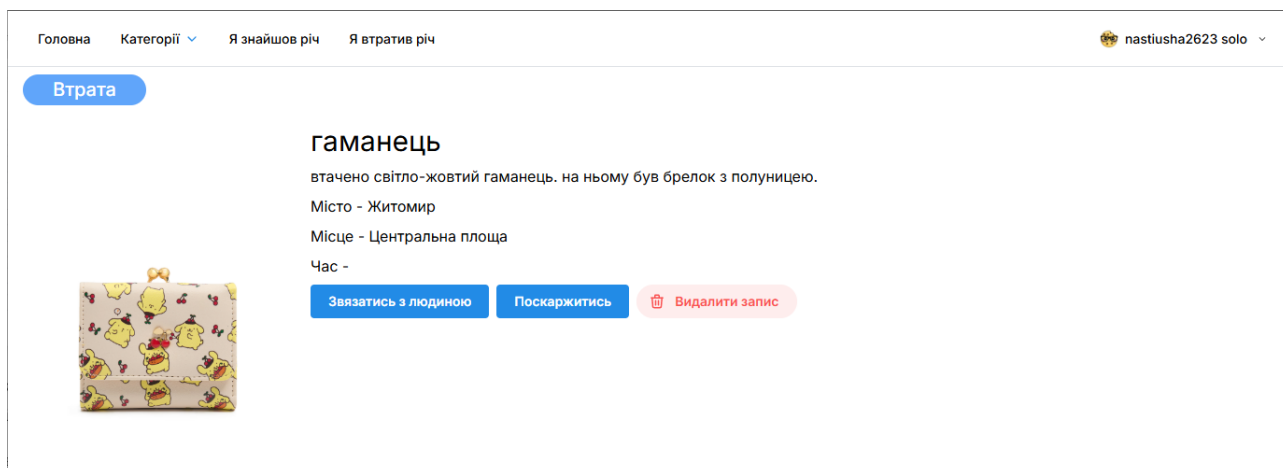


Рис. 3.9 Перегляд власного оголошення

На рисунку 3.10 показано випадаюче меню профілю користувача. Воно з'являється після натискання на ім'я. Це меню надає швидкий доступ до ключових розділів особистого кабінету.

У цьому меню користувач може перейти до розділів «Мої знахідки» та «Мої втрати», що, ймовірно, містять списки доданих оголошень про знайдені або загублені речі. Також є можливість перейти до «Налаштувань» та функція "Вийти" для завершення сеансу.

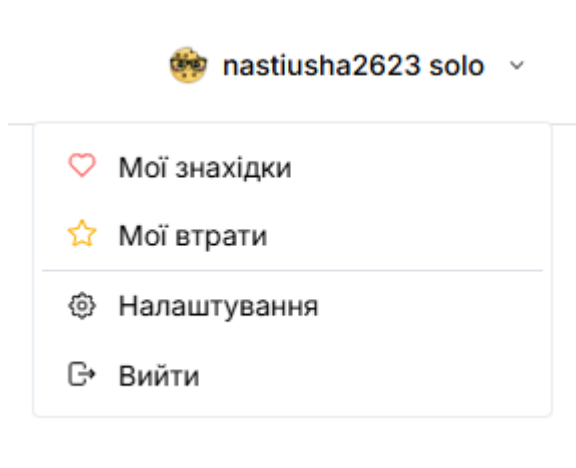


Рис.3.10 Меню профілю користувача

На сторінці «Налаштування»(рис. 3.11) користувач має змогу змінити свої персональні дані такі як ім'я, прізвище, електрона адреса та аватар профілю.

Профіль

Ім'я

Анастасія

Прізвище

Сергієнко

Електронна пошта

nastiusha2623@gmail.com

Аватар

https://lh3.googleusercontent.com/a/ACg8ocIbOsWnQldt9pe-O9zahiVHpQuSbz52xRSBdZcSSCi2nY59ht4=s96-c

Зберегти

Рис. 3.11 Налаштування профілю користувача

3.3. Тестування роботи програмного комплексу

Тестування розробленої системи на працездатність – є невід’ємним процесом під час розробки програмного коду.

Виконаємо перевірки основного функціоналу для знаходження помилок та своєчасного виправлення їх.

Вхід до акаунту

Передумови: користувач попередньо зареєструвався на платформі та зайшов на сторінку входу до акаунта;

Кроки: ввести коректні електронну пошту та пароль

Очікуваний результат: система успішно виконує авторизацію та відкриває користувачу сайт на попередньо відвідуваній сторінці, де у правому верхньому куті відображається ім’я;

Фактичний результат:

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	44
		Фант М.О.				

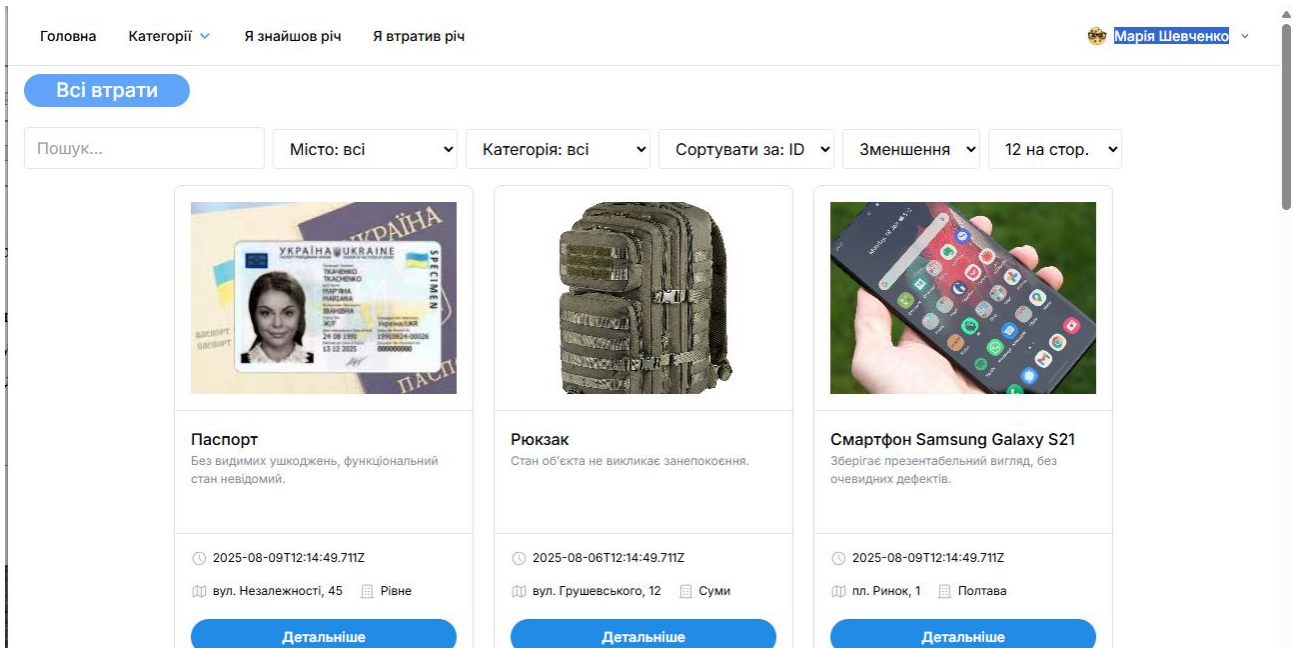


Рис. 3.12 Результат входу до облікового запису

Зміна даних акаунту

Передумови: користувач авторизований та відкрив сторінку «Налаштування»;

Кроки: замінити прізвище з Іванчук на Шевченко та збереження даних.

Очікуваний результат: система повідомляє про успішне збереження та після повторного входу дані оновлюються;

Фактичний результат:

Профіль

Ім'я

Марія

Прізвище

Шевченко

Електронна пошта

maria@example.com

Аватар

https://lh3.googleusercontent.com/a/ACg8oclbOsWnQldt9pe-O9zahiVHpQuSbz52xRSBdZcSSci2nY59ht4=s96-c

Зберегти

Успіх

Профіль оновлено успішно

Рис. 3.13 Зміна даних користувача

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	45
		Фант М.О.				

Сортування знахідок за містом

Передумови: користуван на сторінці перегляду списку знайдених речей;

Кроки: у полі фільтру «Міста» обрати потрібне місто;

Очікуваний результат: система оновлює сторінку та виводить оголошення тільки з обраного міста;

Фактичний результат:

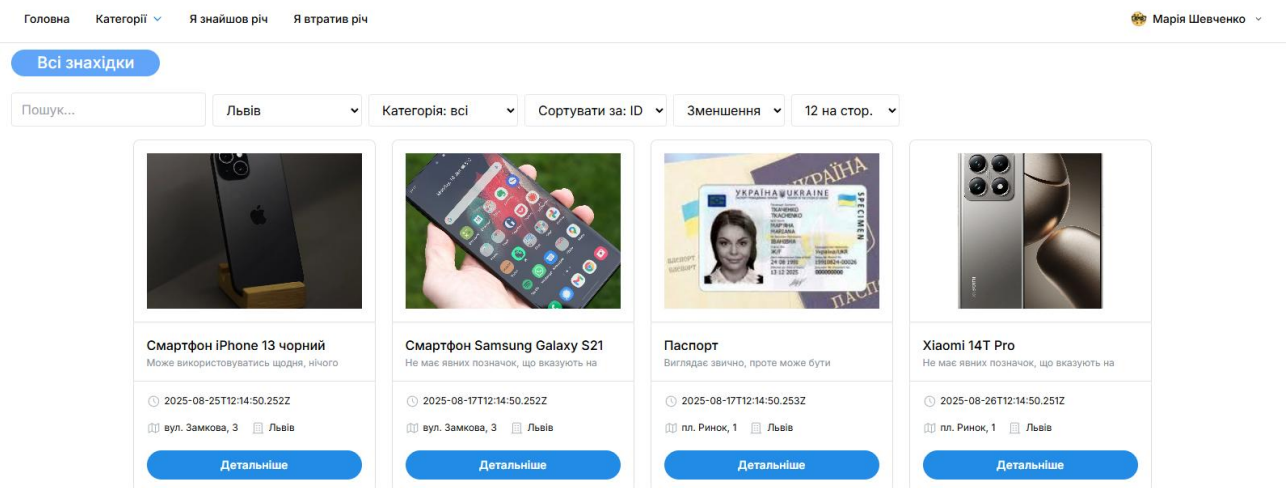


Рис. 3.14 Відсортовані за містом оголошення

У процесі тестування було перевірено весь функціонал платформи. Перевірялися основні сценарії взаємодії користувача із системою, зокрема реєстрація та авторизація, створення, редагування та видалення оголошень, зміна даних облікового запису, пошук і сортування інформації за різними критеріями. За результатами тестування помилок у роботі не виявлено. Усі функціональні можливості виконуються відповідно до заданих вимог. Це свідчить про коректність роботи платформи та готовність її до практичного використання.

Висновки до розділу 3

У цьому розділі розглянуто порядок встановлення та налаштування параметрів системи, а також особливості структури інтерфейсу й користувацької взаємодії з вебплатформою. Наведена діаграма розгортання дозволила наочно представити архітектуру програмного комплексу та продемонструвати зв'язки між клієнтською частиною, сервером і системами зберігання даних.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	46
		Фант М.О.				

Описано ключові елементи інтерфейсу, які забезпечують зручність навігації та простоту використання, включаючи головну сторінку, форми авторизації, функціонал додавання оголошень, персональний кабінет і систему сповіщень. Особливу увагу приділено механізмам обмеження доступу для неавторизованих користувачів, що сприяє підвищенню достовірності інформації на платформі.

Під час тестування перевірені всі основні функціональні можливості платформи, зокрема реєстрація та авторизація користувачів, операції зі створенням, редагуванням і видаленням оголошень, зміна персональних даних, а також пошук і сортування інформації. Результати випробувань показали відсутність помилок у роботі системи, що свідчить про її стабільність та готовність до практичного використання.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						47

ВИСНОВКИ

У рамках випускної кваліфікаційної роботи здійснено повний цикл розробки вебплатформи реєстр пошуку загублених речей. Проєкт охопив аналіз предметної області, архітектурне проєктування, створення інтерфейсу, реалізацію функціональних модулів і тестування готового рішення.

У першому розділі проведено аналіз наявних рішень та визначено ключові функціональні вимоги до системи. Порівняння аналогів дозволило аргументувати вибір архітектури та технологій, зокрема Next.js для фронтенду, tRPC для комунікації між клієнтом і сервером, а також PostgreSQL у парі з Prisma ORM для управління даними.

Другий розділ присвячено архітектурному проєктуванню системи. Було створено UML-діаграми, які відображають сценарії взаємодії користувачів, логіку реєстрації, авторизації, управління оголошеннями, включно з їх пошуком і редагуванням. Розроблена структура бази даних забезпечує інформаційну цілісність та масштабованість системи. Впровадження сучасних бібліотек і інструментів оптимізації сприяло покращенню продуктивності та зручності використання платформи.

Третій розділ описує практичну реалізацію вебплатформи. У ньому представлена структура інтерфейсу, діаграма розгортання для демонстрації клієнт-серверної моделі взаємодії, а також детально розглянуті навігаційні елементи та механізми захисту даних. Під час тестування перевірено основні функції платформи: реєстрацію й авторизацію користувачів, управління оголошеннями, пошук інформації та зручність взаємодії з інтерфейсом. Результати тестування підтвердили стабільну і коректну роботу системи.

Виконана робота демонструє комплексний підхід до розробки сучасної вебплатформи, яка відповідає сучасним стандартам зручності, безпеки та продуктивності. Результати випускної роботи можуть стати основою для подальшого розвитку та вдосконалення платформи для реєстрації та пошуку загублених та знайдених речей.

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						48

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nixon R. Learning MySQL: Get a Handle on Your Data / Robin Nixon. – Sebastopol: O'Reilly Media, 2007. – 456 с.
2. Freeman E., Robson E. Head First React: A Learner's Guide to Building Web Applications Using React and Redux. – O'Reilly Media, 2020. – 500 с.
3. Wieruch R. The Road to React: Your journey to master plain yet pragmatic React.js. – Leanpub, 2020. – 420 с.
4. Buley L. The User Experience Team of One: A Research and Design Survival Guide. – Rosenfeld Media, 2013. – 240 с.
5. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. – 3rd Edition. New Riders, 2014. – 216 с.
6. Nielsen J., Loranger H. Prioritizing Web Usability. – New Riders, 2006. – 288 с.
7. Sommerville I. Software Engineering. – 10th Edition. Pearson, 2015. – 800 с.
8. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. – 2nd Edition. Addison-Wesley, 2005. – 600 с.
9. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – 3rd Edition. Addison-Wesley, 2003. – 150 с.
10. Silva P., Sousa L. User Interface Design Principles for Web Applications. – ACM Conference on Human Factors in Computing Systems, 2019.
11. Ali A., Khan M. Web-Based Systems for Personal Belongings Recovery: A User-Centered Approach. – Journal of Software Engineering, 2022. – Vol. 14, № 3. – P. 45–59.
12. Date C. J. An Introduction to Database Systems. – Pearson, 2003. – 1232 с.
13. Chen L., Zhang D. Web-Based Platforms for AI Image Generation: A Survey. – International Journal of Web Engineering and Technology, 2021.
14. Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks. – IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

15. Welling L., Thomson L. PHP and MySQL Web Development. – Addison-Wesley, 2016. – 1064 с.
16. Duckett J. JavaScript & JQuery: Interactive Front-End Web Development. – Wiley, 2014. – 664 с.
17. Richards M. Software Architecture Patterns. – O'Reilly Media, 2015. – 140 с.
18. Next.js Documentation [Електронний ресурс] – Режим доступу: <https://nextjs.org/docs>
19. React Documentation [Електронний ресурс] – Режим доступу: <https://react.dev/>
20. Mantine UI Documentation [Електронний ресурс] – Режим доступу: <https://mantine.dev/>
21. TanStack Query Documentation [Електронний ресурс] – Режим доступу: <https://tanstack.com/query/latest>
22. NextAuth.js Documentation [Електронний ресурс] – Режим доступу: <https://next-auth.js.org/>
23. Zod Documentation [Електронний ресурс] – Режим доступу: <https://zod.dev/packages/zod>
24. Microsoft Azure Documentation: Cloud Architecture & Deployment [Електронний ресурс] – Режим доступу: <https://learn.microsoft.com/en-us/azure/architecture/>
25. Документація до NestJS [Електронний ресурс] – Режим доступу: <https://docs.nestjs.com>
26. Документація до PostgreSQL [Електронний ресурс] – Режим доступу: <https://www.postgresql.org/docs/>
27. MDN Web Docs: HTML, CSS, JavaScript [Електронний ресурс] – Режим доступу: <https://developer.mozilla.org>
28. Документація для використання Google OAuth [Електронний ресурс] – Режим доступу: <https://developers.google.com/identity/protocols/oauth2>

		Сергієнко А.С.			Житомирська політехніка. 25.121.2.000 - ПЗ	
		Фант М.О.				
						50

29. What Is a REST API? Examples, Uses, and Challenges / Postman Blog
[Електронний ресурс] – Режим доступу: <https://blog.postman.com/rest-api-examples/>
30. Предметно-Орієнтоване Проєктування / Кондратюк С.В. – Режим доступу: <https://conf.ztu.edu.ua/wp-content/uploads/2019/02/47-1.pdf>
31. Вебсервіс Poisk.ua [Електронний ресурс] – Режим доступу: <https://poisk.ua/>
32. Вебсервіс Поверни! [Електронний ресурс] – Режим доступу: <https://verni.com.ua/uk/>
33. Вебсервіс luckfind [Електронний ресурс] – Режим доступу: <https://www.luckfind.me/>

ДОДАТКИ

		Сергієнко А.С			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						52

```

import { trpc } from '@lib/trpc';

export function useUsers() {
  return trpc.user.getAll.useQuery();
}

export function useUser(id: string) {
  return trpc.user.getById.useQuery({ id });
}

export function useCreateUser() {
  const utils = trpc.useUtils();
  return trpc.user.create.useMutation({
    onSuccess: () => {
      utils.user.getAll.invalidate();
    },
  });
}

export function useUpdateUser() {
  const utils = trpc.useUtils();
  return trpc.user.update.useMutation({
    onSuccess: () => {
      utils.user.getAll.invalidate();
    },
  });
}

export function useDeleteUser() {
  const utils = trpc.useUtils();
  return trpc.user.delete.useMutation({
    onSuccess: () => {
      utils.user.getAll.invalidate();
    },
  });
}

export function useLosts(params?: { userId?: string; includeUser?: boolean }) {
  if (params?.userId) {
    return trpc.lost.getByUserId.useQuery({ userId: params.userId });
  }
}

```

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						53

```

    return trpc.lost.getAll.useQuery();
  }

export function usePagedLosses(input?: {
  page?: number;
  limit?: number;
  q?: string;
  userId?: string;
  cityId?: number;
  categoryId?: number;
  sortBy?: 'id' | 'title' | 'time' | 'location';
  sortOrder?: 'asc' | 'desc';
}) {
  return trpc.lost.getPaged.useQuery(input);
}

export function useLost(id: string) {
  return trpc.lost.getById.useQuery({ id });
}

export function useCreateLost() {
  const utils = trpc.useUtils();
  return trpc.lost.create.useMutation({
    onSuccess: () => {
      utils.lost.getAll.invalidate();
    },
  });
}

export function useUpdateLost() {
  const utils = trpc.useUtils();
  return trpc.lost.update.useMutation({
    onSuccess: () => {
      utils.lost.getAll.invalidate();
    },
  });
}

export function useDeleteLost() {
  const utils = trpc.useUtils();
  return trpc.lost.delete.useMutation({
    onSuccess: () => {
      utils.lost.getAll.invalidate();
    },
  });
}

```

```

    },
  });
}

export function useFinds(params?: { userId?: string; includeUser?: boolean }) {
  if (params?.userId) {
    return trpc.find.getByUserId.useQuery({ userId: params.userId });
  }
  return trpc.find.getAll.useQuery();
}

export function usePagedFinds(input?: {
  page?: number;
  limit?: number;
  q?: string;
  userId?: string;
  cityId?: number;
  categoryId?: number;
  sortBy?: 'id' | 'title' | 'time' | 'location';
  sortOrder?: 'asc' | 'desc';
}) {
  return trpc.find.getPaged.useQuery(input);
}

export function useFind(id: string) {
  return trpc.find.getById.useQuery({ id });
}

export function useCreateFind() {
  const utils = trpc.useUtils();
  return trpc.find.create.useMutation({
    onSuccess: () => {
      utils.find.getAll.invalidate();
    },
  });
}

export function useUpdateFind() {
  const utils = trpc.useUtils();
  return trpc.find.update.useMutation({
    onSuccess: () => {
      utils.find.getAll.invalidate();
    },
  });
}

```

```

    });
}

export function useDeleteFind() {
  const utils = trpc.useUtils();
  return trpc.find.delete.useMutation({
    onSuccess: () => {
      utils.find.getAll.invalidate();
    },
  });
}

export function useCategories() {
  return trpc.category.getAll.useQuery();
}

export function useCreateCategory() {
  const utils = trpc.useUtils();
  return trpc.category.create.useMutation({
    onSuccess: () => {
      utils.category.getAll.invalidate();
    },
  });
}

export function useUpdateCategory() {
  const utils = trpc.useUtils();
  return trpc.category.update.useMutation({
    onSuccess: () => {
      utils.category.getAll.invalidate();
    },
  });
}

export function useDeleteCategory() {
  const utils = trpc.useUtils();
  return trpc.category.delete.useMutation({
    onSuccess: () => {
      utils.category.getAll.invalidate();
    },
  });
}

```

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						56

```

export function useCities() {
  return trpc.city.getAll.useQuery();
}

export function useCreateCity() {
  const utils = trpc.useUtils();
  return trpc.city.create.useMutation({
    onSuccess: () => {
      utils.city.getAll.invalidate();
    },
  });
}

export function useUpdateCity() {
  const utils = trpc.useUtils();
  return trpc.city.update.useMutation({
    onSuccess: () => {
      utils.city.getAll.invalidate();
    },
  });
}

export function useDeleteCity() {
  const utils = trpc.useUtils();
  return trpc.city.delete.useMutation({
    onSuccess: () => {
      utils.city.getAll.invalidate();
    },
  });
}

```

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						57

```

interface Props {
  id?: string;
  title?: string | null;
  photo?: string | null;
  description?: string | null;
  location?: string | null;
  time?: string | null;
  city?: {
    name?: string | null;
  } | null;
  userId?: string | null;
}

export function LostCard({ id, title, photo, description, location, time, city,
userId }: Props) {
  const { data: session } = useSession();
  const router = useRouter();

  interface ChatResponse {
    id: string;
  }

  const handleStartChat = async (e: MouseEvent) => {
    e.stopPropagation();

    if (!session?.user) {
      router.push('/api/auth/signin');
      return;
    }

    if (!userId) return;

    try {
      const { data: chat } = await axios.post<ChatResponse>('/api/chats', {
        recipientId: userId,
        lostId: id,
      });
      router.push(`/messages/${chat.id}`);
    } catch (error) {
      console.error('Error starting chat:', error);
    }
  }
}

```

```

};

const badges = [
  { label: time, icon: IconClock },
  { label: location, icon: IconMap },
  { label: city?.name, icon: IconBuilding },
] as const;

const features = badges.map((feature) => {
  if (!feature.label) {
    return null;
  }
  return (
    <Center key={feature.label}>
      <feature.icon size="1.05rem" className={classes.icon} stroke={1.5} />
      <Text size="xs">{feature.label}</Text>
    </Center>
  );
});

return (
  <Card withBorder radius="md" className={classes.card}>
    <Card.Section className={classes.imageSection}>
      <img
        className="w-full h-48 object-cover"
        src={
          photo
          'https://upload.wikimedia.org/wikipedia/commons/1/14/No_Image_Available.jpg'
        }
        alt={' ' + title}
        onError={(e) => {
          const target = e.currentTarget as HTMLImageElement;
          const fallback =
            'https://upload.wikimedia.org/wikipedia/commons/1/14/No_Image_Available.jpg';
          if (target.src !== fallback) target.src = fallback;
        }}
      />
    </Card.Section>

    <Group justify="space-between" className="line-clamp-3" mt="md" h={90}>
      <div>
        <Text fw={500}>{title}</Text>
        {description && (

```

||

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						59

```

        <Text fz="xs" c="dimmed">
            {description}
        </Text>
    })
</div>
</Group>

<Card.Section className={classes.section} h={70} mt="md">
    <Group>{features}</Group>
</Card.Section>

<Card.Section className={classes.section}>
    <Group justify="space-between" wrap="nowrap">
        <Link href={`\lost/${id}`} style={{ flex: 1 }}>
            <Button radius="xl" fullWidth>
                Детальніше
            </Button>
        </Link>
        {userId && userId !== session?.user?.id && (
            <ActionIcon
                variant="filled"
                color="blue"
                size="lg"
                radius="xl"
                onClick={handleStartChat}
                title="Написати повідомлення"
            >
                <IconMessage size={20} />
            </ActionIcon>
        )}
    </Group>
</Card.Section>
</Card>
);
}

```

```

import NextAuth from 'next-auth';
import GoogleProvider from 'next-auth/providers/google';
import CredentialsProvider from 'next-auth/providers/credentials';
import { JWT } from 'next-auth/jwt';
import * as jsonwebtoken from 'jsonwebtoken';
import { prismaAdapter } from '@lib/auth/prisma-adapter';
import { prisma } from '@lib/prisma';
import bcrypt from 'bcryptjs';

const handler = NextAuth({
  secret: process.env.NEXTAUTH_SECRET,
  providers: [
    GoogleProvider({
      clientId: process.env.NEXT_PUBLIC_GOOGLE_CLIENT_ID!,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET!,
      allowDangerousEmailAccountLinking: true,
    }),
    CredentialsProvider({
      name: 'Email and Password',
      credentials: {
        email: { label: 'Email', type: 'email', placeholder:
'example@example.com' },
        password: { label: 'Password', type: 'password' },
      },
      async authorize(credentials) {
        console.log({ CREDENTIALS: credentials });
        if (!credentials?.email || !credentials?.password) {
          return null;
        }

        try {
          const user = await prisma.user.findUnique({
            where: { email: credentials.email },
          });

          if (!user || !user.password) {
            console.log('User not found or no password');
            return null;
          }
        }
      }
    })
  ]
});

```

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						61

```

        const isPasswordValid = await bcrypt.compare(credentials.password,
user.password);

        if (!isPasswordValid) {
            console.log('Invalid password');
            return null;
        }

        return {
            id: user.id,
            email: user.email,
            name: user.name || `${user.first_name} ${user.last_name}`,
            image: user.image,
        };
    } catch (error) {
        console.error('Error in credentials authorization:', error);
        return null;
    }
},
    }),
],
adapter: prismaAdapter,
theme: {
    colorScheme: 'auto',
},
session: { strategy: 'jwt' },
jwt: {
    encode: ({ secret, token }: any) => {
        return jsonwebtoken.sign(token!, secret, {
            algorithm: 'HS256',
        });
    },
    decode: async ({ secret, token }: any) => {
        const decodedToken = jsonwebtoken.verify(token!, secret, {
            algorithms: ['HS256'],
        });
        return decodedToken as JWT;
    },
},
callbacks: {
    async jwt({ token }: any) {
        return {
            token,

```

```

    'https://hasura.io/jwt/claims': {
      'x-hasura-allowed-roles': ['user'],
      'x-hasura-default-role': 'user',
      'x-hasura-role': 'user',
      'x-hasura-user-id': token.sub,
    },
  };
},
session: async ({ session, token }: any) => {
  if (session?.user) {
    session.user.id = token.sub!;
  }
  return session;
},
});

export { handler as GET, handler as POST };

```

		Сергієнко А.С.			Житомирська політехніка.25.121.2.000 - ПЗ	
		Фант М.О.				
						63